NO. 20 JULY 1984 $3.00

THE MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING™

0 74470 12385 07

From the editors of
A.N.A.L.O.G. Computing

$14.95

THE A.N.A.L.O.G COMPENDIUM

The best ATARI® Home Computer Programs from the first ten issues of A.N.A.L.O.G. Computing Magazine.

All Compendium programs are available on DISK.

The **ANALOG Compendium** is available at selected book and computer stores, or you can order it direct. Send a check or money order for $14.95 + $2 shipping and handling to: **ANALOG Compendium, P. O. Box 615, Holmes, PA 19043**.

Or you can order by phone with MasterCard or VISA. Call toll free: 1-800-345-8112 (in PA, call 1-800-662-2444). **For orders outside the U.S., add an additional $5 air mail, $2 surface.**

CIRCLE #154 ON READER SERVICE CARD.

NO. 20 JULY 1984

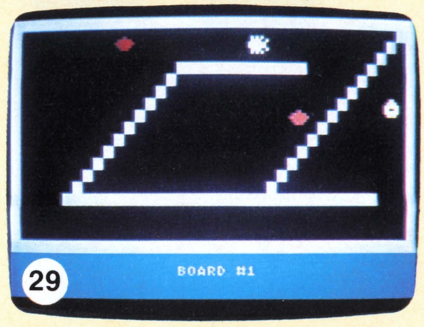
THE MAGAZINE FOR ATARI® COMPUTER OWNERS

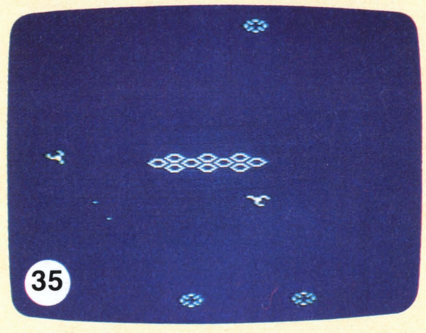
ANALOG COMPUTING™

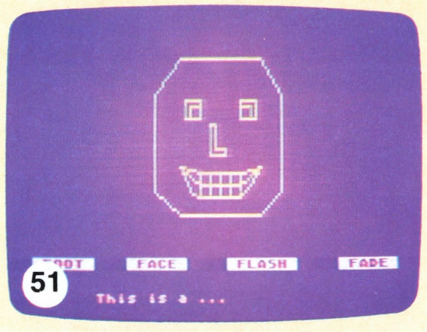
FEATURES

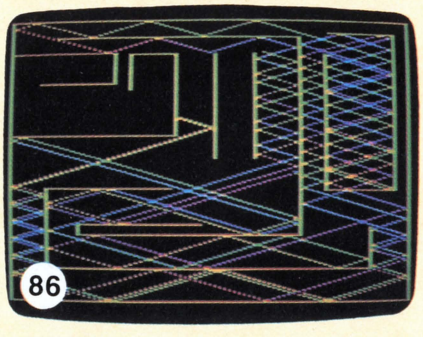
PRODUCT REVIEWS

COLUMNS







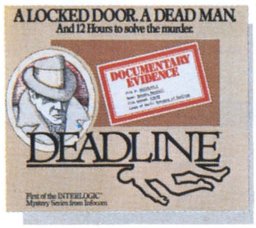


ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by ANALOG Magazine Corp., 565 Main Street, Cherry Valley, MA 01611. Second-class postage paid at Worcester, MA and additional mailing offices.


Your greatest challenge lies ahead–and downwards.
ZORK I
INFOCOM

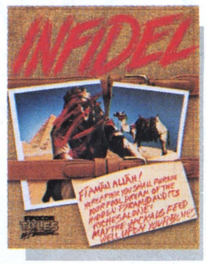

The next step downward is danger.
ZORK II
INFOCOM

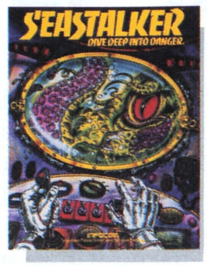

It all comes down to this.
ZORK III
INFOCOM


A LOCKED DOOR. A DEAD MAN.
And 12 Hours to solve the murder.
DOCUMENTARY EVIDENCE
DEADLINE
First of the INTERLOGIC™ Mystery Stories from Infocom


Get INSIDE!
The WITNESS
SOMEBODY'S GOING TO TAKE THE DEEP SIX!


STARCROSS
INFOCOM





THE INCOMPLETE WORKS OF INFOCOM, INC.

Incomplete, yes. But it's not just because we're always bringing out new stories in the Infocom interactive fiction collection. Nor is it simply due to the fact that with all the writing and re-writing, honing and perfecting that we put into every one of our stories, our work is seemingly never done.

The real reason is: an Infocom work of fiction can never be complete until you become a part of it.

You see, as hard as we work at perfecting our stories, we always leave out one essential element—the main character. And that's where you enter in.

Once you've got Infocom's interactive fiction in your computer, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities), and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, not passive. The course of events is shaped by the actions you choose to take. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

In other words, only you can complete the works of Infocom, Inc. Because they're stories that grow out of your imagination.

Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.

INFOCOM™

Infocom, Inc., 55 Wheeler Street, Cambridge, MA 02138

For your: Apple II, Atari, Commodore 64, CP/M8", DECmate, DEC Rainbow, DEC RT-11, IBM PC* and PCjr, KAYPRO II, MS-DOS 2.0*, NEC APC, NEC PC-8000, Osborne, Tandy 2000, TI Professional, TI 99/4A, TRS-80 Models I and III.

*Use the IBM PC version for your Compaq, and the MS-DOS 2.0 version for your Wang or Mindset.

CIRCLE #102 ON READER SERVICE CARD.

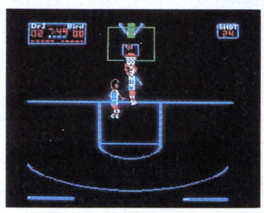
SOFTWARE ARTISTS?

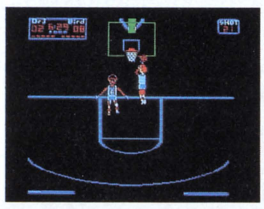
How we got this year's hottest sports game out of two rather inexperienced designers.

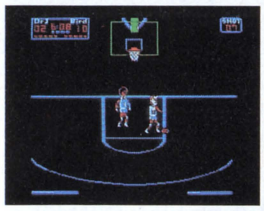
TO MAKE THE FIRST BASKETBALL PROGRAM that feels like the real thing, it helps to start with two guys who know what the real thing feels like.

Enter Larry Bird and Julius Erving. Bird – the hustler, the strong man, deadly from outside. Erving – The Doctor, maybe the most explosive player in the history of the game.

We talked to them, photographed them in action, studied their moves and their stats and their styles. Then we set out to create on computer disc an event which may never happen in real life. We put the two of them together on a dream court of light, for an electronic afternoon of one-on-one.





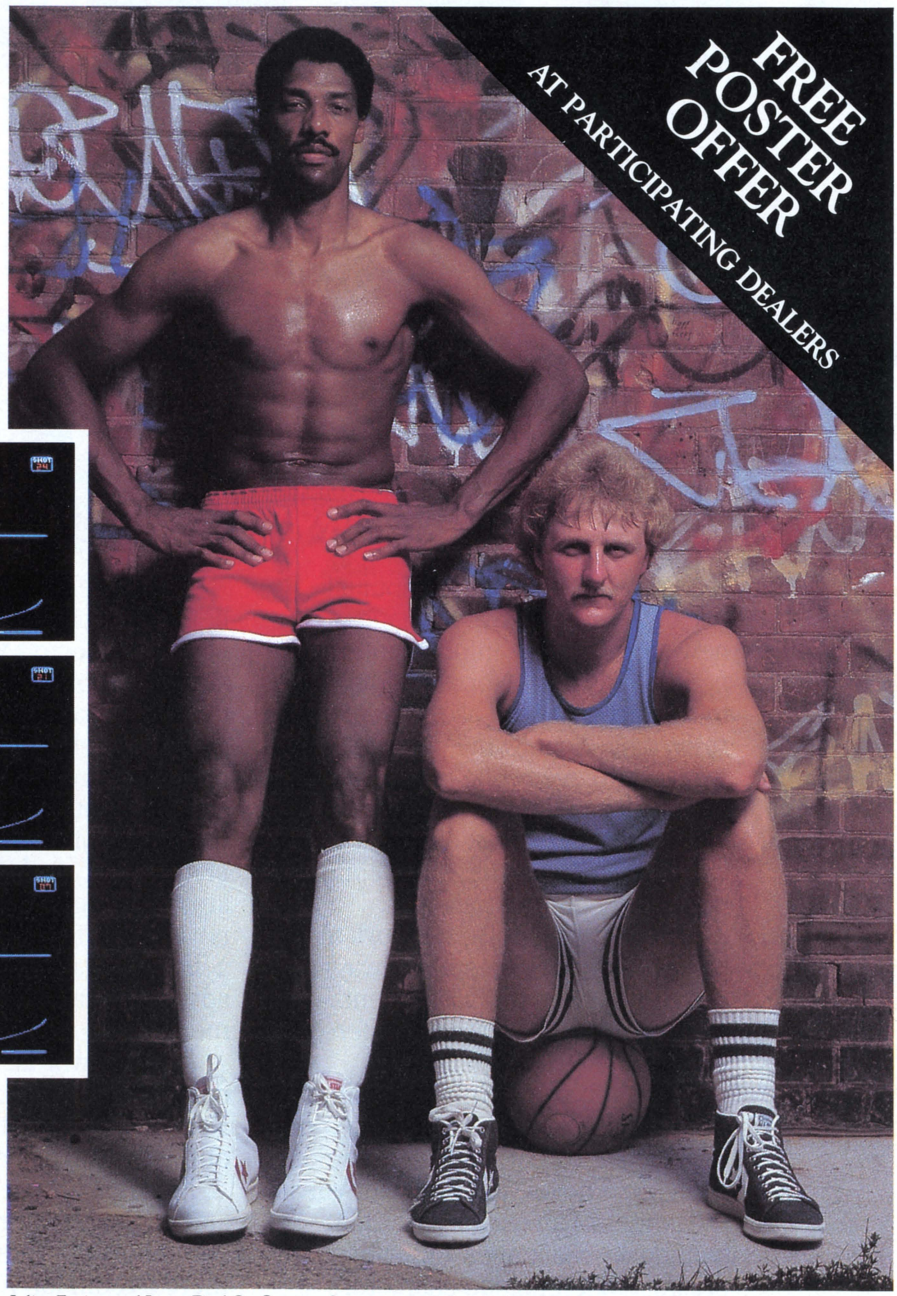


It wasn't easy. When they talked, we listened. When they criticized, we made big changes. When they gave suggestions, we took them.

And it shows. This thing is absolutely uncanny. You actually take on all the skills and characteristics of Bird or The Doctor – their own particular moves, shooting abilities, even strength and speed.

You'll meet with fatigue factors, hot and cold streaks, turnaround jump shots, and 360-degree slam dunks. But there's some whimsy in here, too – a funny referee, a shattering backboard, even instant replay.

It's called *Julius Erving and Larry Bird Go One-on-One.*™ You're Bird. Or you're The Doctor. And that's the last decision you'll have plenty of time to make.



Julius Erving and Larry Bird Go One-on-One is now available on diskette for Apple II, II+, and IIe computers. Apple is a registered trademark of Apple Computer. To find out more about Electronic Arts and its products, write us at 2755 Campus Drive, San Mateo, CA 94403 or call (415) 571-7171.

NOW AVAILABLE FOR ATARI HOME COMPUTERS

ELECTRONIC ARTS™

CIRCLE #103 ON READER SERVICE CARD.

ANALOG STAFF

Editors/Publishers
MICHAEL DES CHENES
LEE H. PAPPAS

Managing Editor
JON A. BELL

Production Editor
DIANE L. GAW

Contributing Editors
JOEL GLUCK
BRADEN GRIFFIN, M.D.
TONY MESSINA

Art Director
BOB DESI

Contributing Artists
GARY LIPPINCOTT
LINDA RICE

Technical Division
CHARLES BACHAND
TOM HUDSON
KYLE PEACOCK

Advertising Manager
MICHAEL DES CHENES

Distribution
PATRICK J. KELLEY

Production/Distribution
LORELL PRESS, INC.

Contributors
GORDON L. BANKS
RAY BERUBE
BOB CURTIN
DONALD E. GLOVER
RON HODGE
DAVID KARP
ARTHUR LEYENBERGER
LARRY W. LINSON
DAVID PLOTKIN
SAM WILEY

ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by **ANALOG** Magazine Corp., 565 Main Street, Cherry Valley, MA 01611, Tel. (617) 892-3488. Second-class postage paid at Worcester, MA and additional mailing offices. POSTMASTER: Send address changes to **ANALOG COMPUTING**, P.O. Box 615, Holmes, PA 19043.

Program listings should be provided in printed form. Articles should be furnished as typed copy in upper and lower case with double spacing. By submitting articles to **ANALOG COMPUTING**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply self-addressed, stamped envelope. U.S.A. Newstand distribution by Eastern News Distributors, Inc., 111 Eighth Ave., New York, NY 10011.

READER COMMENT

Beware the dreaded modem.

I wear two hats. One belongs to the owner of an Atari 800; the other to an electronics engineer with a large, independent telephone company. As an Atari owner, I am intrigued by the doors that open when a modem is attached. Suddenly, my computer can talk to another computer across the country. In fact, if I assert myself after all the digital handshaking is complete, I can actually talk to the humanoid at the other end!

This is all very nice, until I put on my telephone engineer's hat. Then I'm appalled at the runaway abuse of the telephone network by some of the modem designers and users. Half-baked equipment, some of it very expensive, is flooding the market. Equally half-baked documentation accompanying it promises features that the telephone network cannot possibly guarantee. As a result, the deceived modem user turns against the telephone company, complaining that his modem sometimes works, sometimes produces errors; or it doesn't work at home, but works at his friend's house; or he can place a call to a bulletin board 1000 miles away, but not to another board 100 miles away, etc.

First of all, the telephone network is designed to supply a voice grade service. And "voice" means just that—not a private analog data circuit, which is what too many of us seem to expect. All telephone companies meet national performance standards on voice quality. These standards are monitored at the federal level and by the state Public Service Commission. Because the voice grade network performs so well, some modem designers and users make false assumptions about what they can demand from it. They think of it as a network that exists only to transmit their two sinewave tones in the voice band, without error.

In particular, they are very hazy about such things as: A. Data signal send levels; B. Switched network losses from call to call, that depend on the path chosen by a central computer in another part of the country perhaps; C. Random noise appearing on the telephone pair; D. Hum appearing on the line (courtesy of the local or distant electric utility); E. Impulse noise level, duration and frequency (these can look like data signals); F. Ringing frequencies and voltages, which are often as much as 100 volts AC; G. Echo amplitude and delay from satellite links; and H. The so-called "ringback" tone. (This is what you hear while you are waiting for your party to pick up his phone. Note: you do not hear his phone ringing. His phone may, in fact, be disconnected. You will still hear the ringback. So don't complain that you heard his phone ringing — you didn't.)

Armed with this lack of understanding, a modem designer can easily turn out a marginal design. The modem's performance will then vary with time and circumstance, in a manner beyond the designer's ken. Nevertheless, he sells it to the innocent consumer, whose

(continued on page 8)

ALOG COMPUTING **PRESENTS**

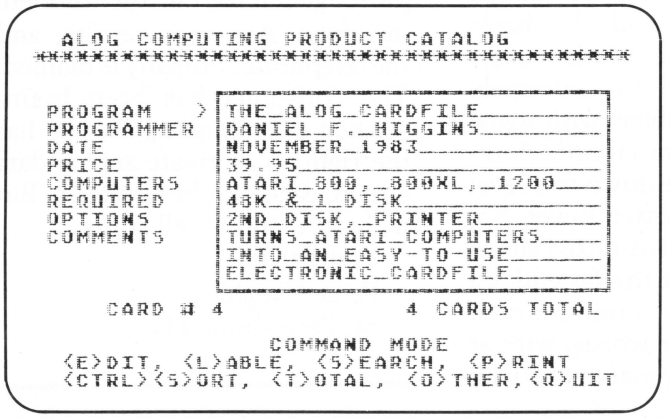
HIGH QUALITY PROGRAMS AT AFFORDABLE PRICES FOR ALL ATARI COMPUTERS WITH 48K AND 1 DISK DRIVE.

SUMMER SPECIAL — BUY TWO & GET ONE FREE!*

*** WHEN ORDERING DIRECT FROM ALOG**

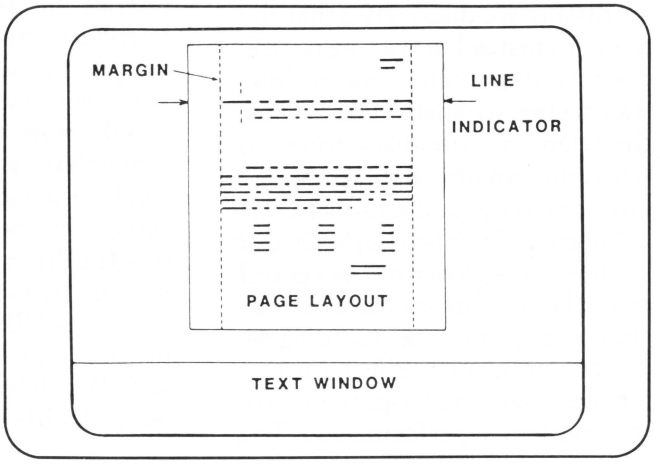
★★★★NEW!★★★★
THE ALOG CARDFILE

TURNS YOUR ATARI INTO AN ELECTRONIC CARD FILE. SEARCH & SORT ON ANY LINE. FORMATTED PRINT AND NUMERIC FIELDS MAKE THE CARDFILE IDEAL FOR YOUR EXPENSE RECORDS, INVENTORIES, AND RECEIPES.



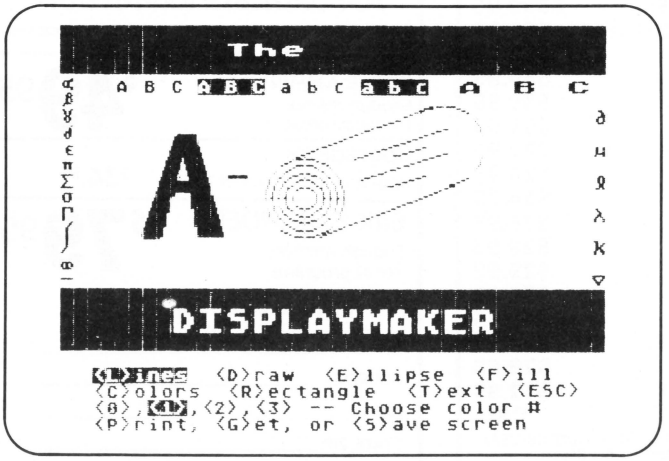
THE ALOG PAGEWRITER

TURNS YOUR ATARI AND 80 COLUMN PRINTER INTO AN EASY-TO-USE ELECTRONIC TYPEWRITER. THE ENTIRE PAGE LAYOUT IS DISPLAYED GRAPHICALLY WHILE YOU TYPE. IDEAL FOR SIMPLE WORD PROCESSING TASKS.



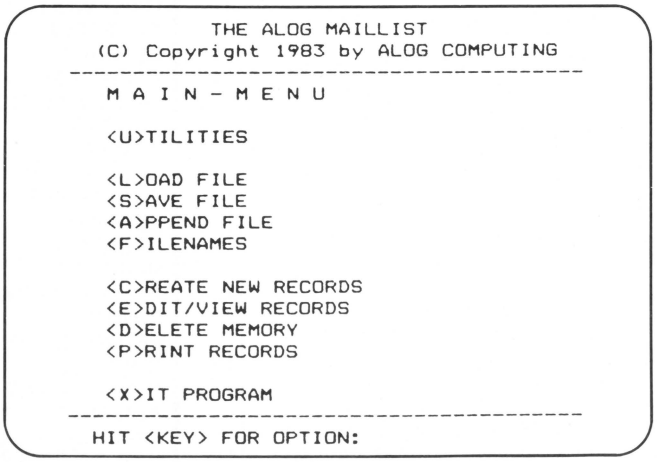
THE ALOG DISPLAYMAKER

MIXES COLOR GRAPHICS AND THREE TEXT SIZES ON THE SAME SCREEN. BUILT IN CHARACTER EDITOR AND SHAPE DRAWING ROUTINES. GRAPHICS DUMP TO MANY PRINTERS. GREAT FOR MAKING GRAPHS, CHARTS, AND HAVING FUN.



THE ALOG MAILLIST

IS AN EASY-TO-USE, MENU-DRIVEN DATA BASE PROGRAM DESIGNED FOR MAILING LISTS AND PRINTING LABELS. FAST 3 KEY SORT AND MULTI-LEVEL SEARCH ON ANY LINES. PRINT FORM LABELS 1 TO 2 INCHES HIGH.



All programs are only $39.95 each including tax. To order direct send check or money order to **Alog Computing, 1040 Veronica Springs Road, Santa Barbara, CA 93105.** We pay shipping. For information, credit card orders or C.O.D., call our distributor: **COMSTAR (805) 964-4660.**

Dealer and distributor inquiries invited.

(ATARI is a trademark of Atari, Inc.)

CIRCLE #105 ON READER SERVICE CARD.

dreams are soon shattered.

A specific case, if you will. The customer had just purchased a high quality portable computer (an Osborne), with an integrated modem and audo-dialer. It worked well from his residence, and from his friend's house, but not from either of his two business lines. I spent much time and money looking for faults in the telephone plant. The cable performance was in limits, but we were loathe to blame a prestigious computer manufacturer. We should have, because eventually we discovered that the modem, instead of waiting for the distant computer to come on-line, was trying to shake hands with the telephone ring-back tone. So when this ring-back tone stopped after one second, the modem disconnected. The customer's place of business was closer to the central office than was his residence, and the stronger ring-back was confusing the poor old modem. In effect, the better the telephone loop, the worse the modem's performance. The customer was extremely embarrassed by it all, and told me he would get rid of the computer.

There is an industry standard governing the maximum tone amplitude that may be placed on a telephone line by a modem. This level is -9dBm. (i.e., 9dB below 1 milliwatt.) Sometimes, a modem user who has trouble getting through opens up his unit, finds the LEVEL control, and cranks it up to the maximum. He knows nothing about crosstalk. He may care even less about the interference he is causing to other telephone users. He is determined to get through, and the heck with everyone else. Modem users should note that it is extremely rare to have a problem with signal level on a modem circuit. A receiver can operate down to a level of -40dBm, and often lower. So, starting from a transmitted signal level of -9dBm at the far end, all tones will arrive at a minimum of -25dBm. In other words, with at least a 15dB safety margin. So why crank up the send level and interfere with someone else?

Other problems have occurred because some modem designers fail to guard against impulse noise on the line . . . may not have given it a moment's thought, even. Impulse noise is caused by household appliances, power surges, CB or ham radio transmitters turning on and off, or lightning. An impulse is broadband, and some part of it will pass through any mark-space tone filter system. A poorly designed guard circuit will allow this impulse to false-trigger the pulse generator at the modem interface. An error results, and the telephone company is blamed. The consumer has been brainwashed into thinking that he has the right to a private analog data circuit, each time he hooks up. But he hasn't, and we all suffer.

Yours truly,
Peter Vaughan
St. Petersburg, FL

MOVING?

DON'T MISS A SINGLE ISSUE.

Let us know your new address right away. Attach an old mailing label in the space provided and print your new address where indicated.

QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

☐ New subscription. Please allow 4-8 weeks for your first copy to be mailed.
☐ Renewal subscription. Please include a current address label to insure prompt and proper extension. ☐ 1-year $28.00. This rate limited to the U.S.A. and its possessions. ☐ Payment enclosed or ☐ Bill me.

ATTACH LABEL HERE

(If label is not handy, print OLD address in this space.)

MAIL TO: ANALOG COMPUTING
P.O. Box 615, Holmes, PA 19043

Name _____

Address _____

City _____ State _____ Zip _____

The Winners

ANALOG's staff picks their all-time favorite games.

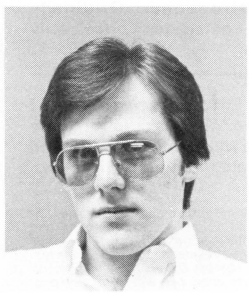
The idea for this article sprang from a reading of *Road and Track* and *Motor Trend*. Occasionally, *R&T* lists what cars their staff is currently driving, and annually, *MT* picks their "Car of the Year" awards. Let me first say that these few pages were done as more or less a fun-thing-to-do, and, in actuality, our top game list probably changes on a daily basis.

Reasons for a favorite game vary greatly, and it's frequently difficult to remember some old favorites when flashy new ones pop up. However, we've tried to overcome this, and, indeed, games from the entire Atari computer lifespan appear on the list. To give older games a fair chance, we took an "after the dust settles" outlook on the newer ones that have recently come to market.

Old favorites like **Star Raiders** and **Missile Command** have what it takes to hold a player's interest and keep you coming back for more self torture. And those in the middle, two years old or so, are still fond in (some of) our memories, such as **Pac-Man** and **Threshold**. But there's no question that some late bloomers, like **MiG Alley Ace** and **Boulder Dash**, have caught our eye(s).

The nominations.

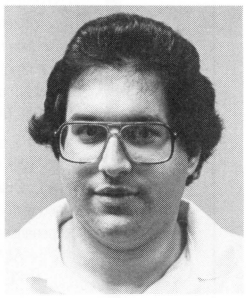
The editorial staff of **ANALOG** was asked to select their seven favorite games of all time. And believe us, the **ANALOG** game library is so vast, it must have shifted the earth's magnetic poles. The final list of chosen games consists of thirty-three different titles from eighteen manufacturees. Except for Atari, none of the other software companies has more than two or three games on the list, and most have only one.



Jon A. Bell

1. **Star Raiders**
2. Archon
3. Miner 2049er
4. Missile Command
5. Pac-Man
6. Threshold
7. Breakout

***Star Raiders** on a Kloss Video Beam, room lights off, the FINAL COUNTDOWN score on 7. Sink back into your Recaro desk chair and let the photons etch out your brain.*



Lee H. Pappas

1. **Star Raiders**
2. Missile Command
3. Lode Runner
4. MiG Alley Ace
5. Krazy Shootout
6. Breakout
7. Boulder Dash

***Star Raiders** — Commander Level: Star Commander Class 1, no shields used the entire game, 54 Zylons destroyed. April 20th, 1984. That's it, that's all.*

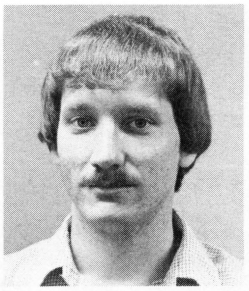
Michael DesChenes

1. **MiG Alley Ace**
2. Missile Command
3. Castle Wolfenstein
4. River Raid
5. Bruce Lee
6. Silicon Warrior
7. The Return of Heracles

I don't enjoy sitting alone playing a one-person computer game. Multiple-player interactive games are the only ones that will ever make it on my list of all-time favorites.

Each reviewer was also asked to quote on his favorite game, the video game business in general, or today's weather. Finally, we tabulated the results of chart frequency to show Atari leading the way with **Star Raiders** (six votes), followed closely by **Missile Command** and newcomer **MiG Alley Ace**.

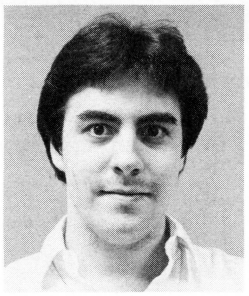
This survey proves nothing, except that taste in games differs . . . from hard-core, blast-'em-to-bits all the way to "Drink Magic Potion." And, finally, the games we've chosen are the best of the best, so your software collection will suffer no ill-effects if you dash out to buy any of these products. □



Tom (HUD) Hudson

1. **Star Raiders**
2. Archon
3. Boulder Dash
4. Miner 2049er
5. Missile Command
6. MiG Alley Ace
7. Donkey Kong

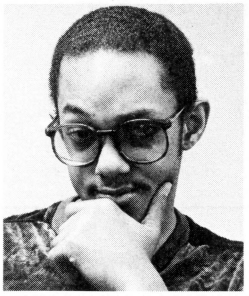
The ultimate test of a truly good game is its lifespan — most last a couple weeks or less. Others, like ***Star Raiders****, are still fun five years after their introduction.*



Pat Kelley

1. **Archon**
2. MiG Alley Race
3. Orc Attack
4. Operation Whirlwind
5. Choplifter
6. Star Raiders
7. Sub Commander

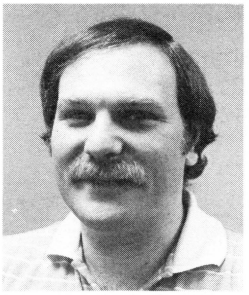
In the cutthroat world of games, EA's ***Archon*** *is a real killer. What else can I say about a game I've devoted over 100 hours of my life to?*



Kyle Peacock

1. **Star Raiders**
2. Encounter
3. Missile Command
4. Pole Position
5. Starcross
6. Choplifter
7. M.U.L.E.

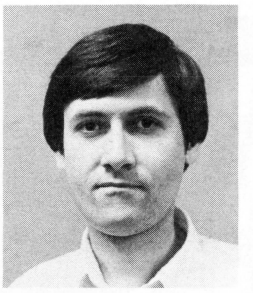
Encounter *does for my visual senses what* ***Starcross*** *does for my unending quest for the stars.*



Tony Messina

1. **Star Raiders**
2. Wizard of Wor
3. Encounter
4. MiG Alley Ace
5. M.U.L.E.
6. Shamus
7. Agent USA

To computer-illiterate jugheads, ***Star Raiders*** *is just another video game. To the more perceptive,* ***Star Raiders*** *is the ultimate simulation from a $200 graphics box. Fifteen years ago, you would have been playing it on a half-million dollar machine — not in civilian hands.*

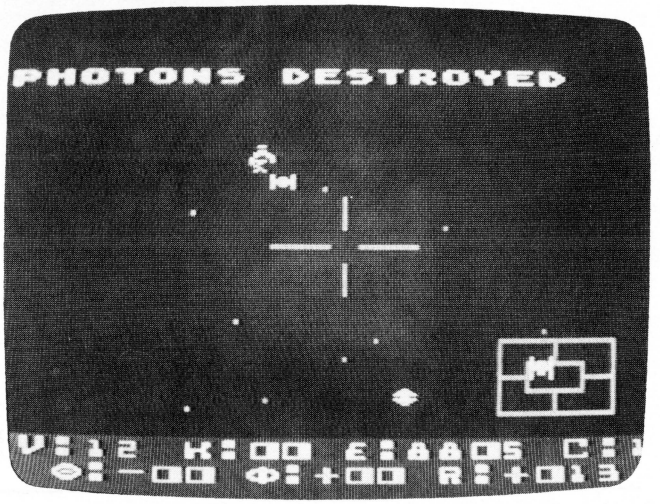


Charles Bachand

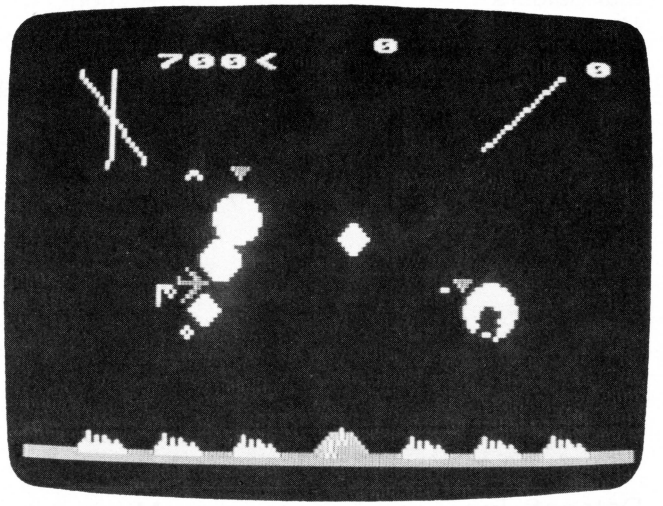
1. **Gateway to Apshai**
2. Ali-Baba
3. Gruds in Space
4. Archon
5. Miner 2049er
6. Wayout
7. Zork I

I must be into self-torture, for my favorite games tend to be the most frustrating. And for sheer masochism, my vote goes to ***Gateway.***

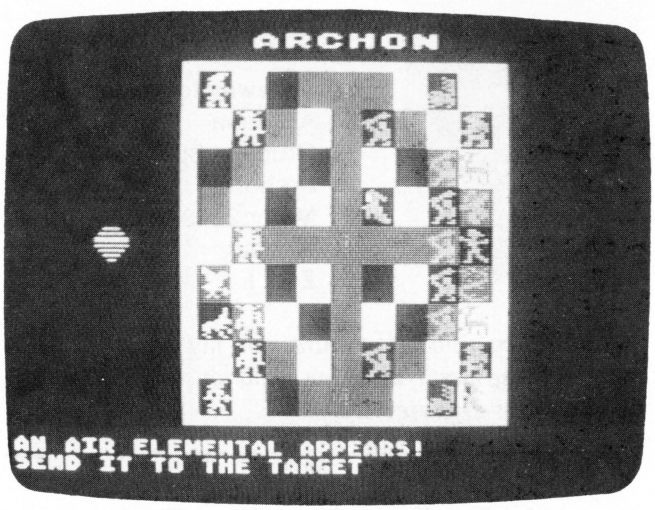
ANALOG's Favorites.



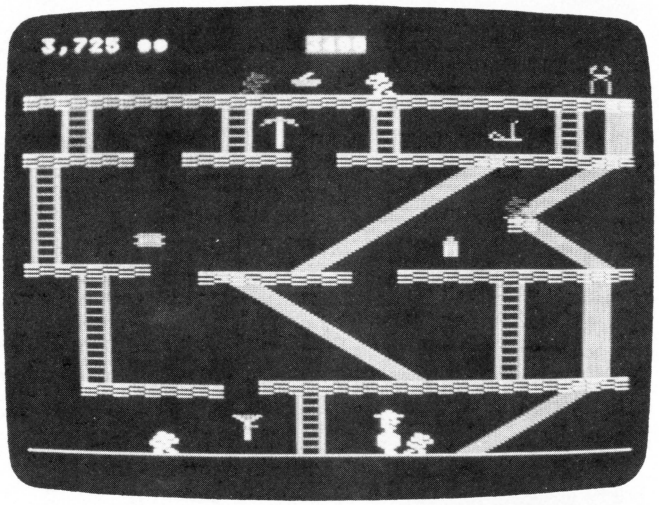
Star Raiders.



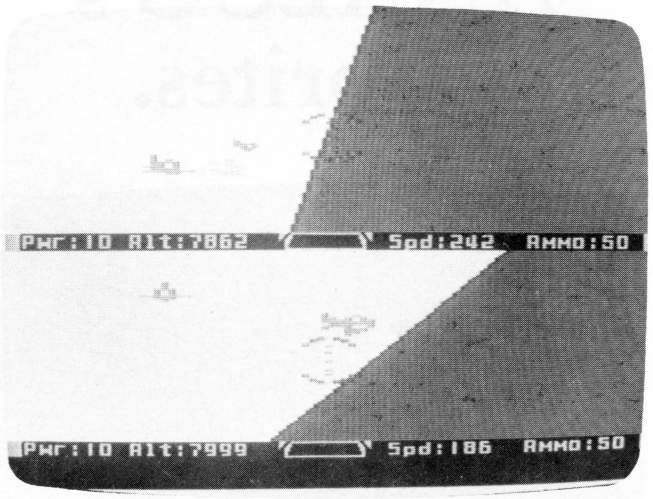
Missile Command.



Archon.



Miner 2049er.



MiG Alley Ace.

The Top Five

1. **Star Raiders — Atari**
2. Missile Command — Atari
   MiG Alley Ace — Microprose
   (Tie)
3. Archon — Electronic Arts
4. Miner 2049er — Big Five

Manufacturers Listing

Atari, Inc.
Star Raiders
Missile Command
Pole Position
Pac Man
Donkey Kong
Breakout

Big-Five Software
Miner 2049er

Broderbund
Choplifter
Loderunner
Operation Whirlwind

CBS Software/Entertainment
Krazy Shootout
Wizard of Wor

Datasoft, Inc.
Bruce Lee

Electronic Arts
Archon
M.U.L.E.

EMI Software
Orc Attack
Sub Commander

EPYX
Gateway to Apshai
Silicon Warrior

First Star Software
Boulder Dash

Infocom
Starcross
Zork 1

Microprose
MiG Alley Ace

MUSE
Castle Wolfenstein

Quality Software
Ali-Baba
The Return or Heracles

Scholastic Winners
Agent USA

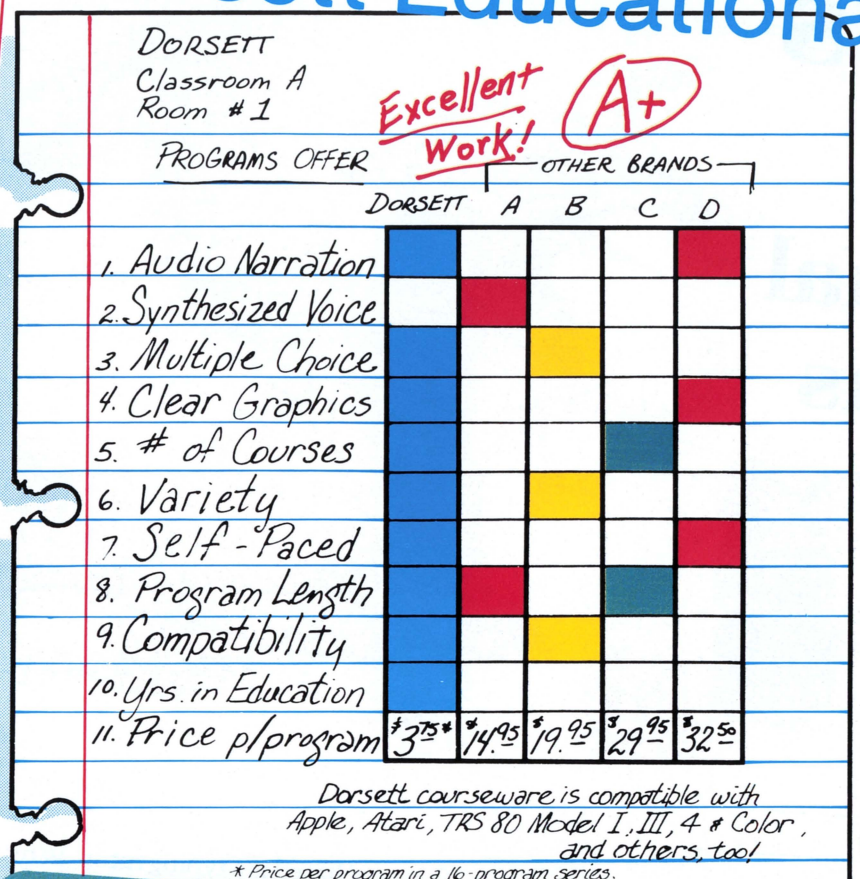
Sierra/On-Line
Threshold

Sirius Software
Gruds in Space
Wayout

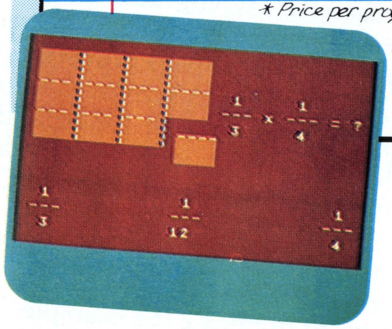
Synapse Software
Encounter
Shamus

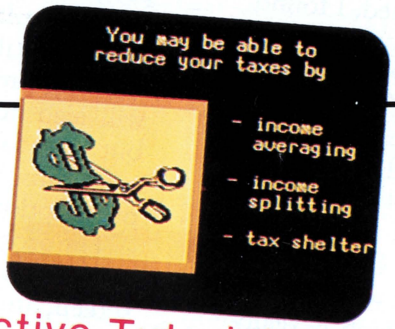
For Your Atari Computer
TRS 80 Color, I, III & 4 or Apple Computer

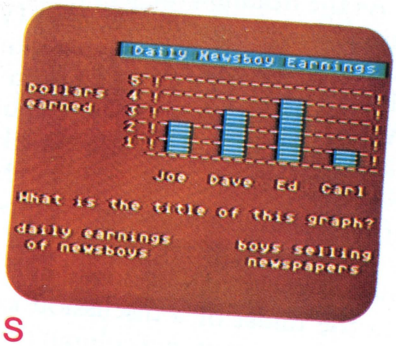
Dorsett Educational Software Passes the Cost-Efficiency Test!



For computer assisted instruction, our software more than passes the test...it EXCELS! With over 1000 educational programs designed specifically to concentrate learning for all users, from kindergarten level to advanced individual studies, from slow learners to exceptionally bright students, you'll get positive results with our quality, economically-priced courseware.







Interactive Tutorial Programs

Over 1000 Programs with Full Time Audio Narration, Pictures & Text!

We're Your Educational Software Source

Course	No. of Programs
Reading	256
Mathematics	128
Comprehension	48
History	32
Algebra	16
Spelling	16
Government	16

16 Programs in each of the following:
Carpentry - Electronics - Health Services
Office Skills - Statistics - First Aid/Safety
Economics - Business - Accounting
Psychology
AND MANY MORE!
Dealer inquiries welcome

For your Atari 400/600/800/1200, you will need the Atari Cassette Recorder and the Dorsett 4001 Educational Master Cartridge, **$9.95**. For your Apple II, you will need the Dorsett M402 T/T plug-in board, **$99.00**, and the M401 stereo cassette player, **$79.00**. All programs listed are available for TRS 80, I, III, 4, which require the M203 speaker converter, **$99.00**, and 401 stereo cassette player, **$79.00**.

$59.90 for an album containing a 16-program course (8 cassettes with 2 programs each at **$3.75** per program). **$8.80** for a 2-program cassette.

Send for a catalog of over 1000 programs for Atari, TRS 80, Apple, etc.

For more information, or to order call:

TOLL FREE 1-800-654-3871
IN OKLAHOMA CALL (405) 288-2301

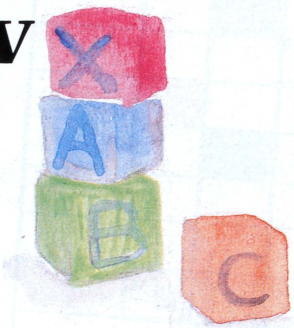
DORSETT
Educational Systems, Inc.
Box 1226, Norman, OK 73070

MasterCard
VISA

CIRCLE #112 ON READER SERVICE CARD.

Griffin's Lair

Educational Programs Review



by Braden E. Griffin, M.D.

As the deadline for issue 19 approached, I found myself so overwhelmed at work (you know, Doctor-stuff . . . saving lives . . .) that I knew I would not be able to complete my column in time. The staff at **ANALOG** is so considerate about extending deadlines for me that I feel a bit like Bridie Murphy. Anyhow, I called the co-boss (Michael) and asked if it would be a problem if I missed an issue. After some remark to the effect that I might be responsible for a second Jonestown, my request was granted. In the midst of a discussion as to which is really more important, my column or the lives of tiny premature infants on respirators, sprinkled with incredibly ghoulish suggestions as to the dedication of my column *in absentia*, I was asked what the theme was to have been. I said that I was reviewing a number of typing programs. To which Michael dryly responded, "You mean programs that teach you how to type faster and get things done on time?" *Touche!*

One of the major reasons for buying a home computer is the ability to manage the written word. School reports, term papers, correspondence, articles for **ANALOG** and "The Great American Novel" are but a few of the many uses of this important resource. Typing is certainly an integral part of this process, and the more facile this skill, the less drudgery in the writing. Typing instruction seems to be a natural for computer education. One of the first educational programs often purchased is a typing tutor. Why not? A most utilitarian skill, typing is a long-term asset, at least until the voice-activated word processors of the future arrive. One wonders if this new technology will lead not only to the atrophy of typing skills, but also to the demise of penmanship. Someday, everything written by hand may look like an M.D.'s prescription.

The programs reviewed this month are basically the same. Letters, words, phrases, etc., appear on the screen, and one tries to duplicate the example. The examples may take the form of an enemy attack, the destruction of which requires rapid and accurate reproduction. Whatever the method, the aim is to teach one to enter characters quickly from the keyboard without looking at the keys. Some are a little more fun, others more classically structured, but all will achieve their purpose with adequate motivation. Displaying the input at eye level, augmented by the use of graphics, is a definite advantage of computer instruction. A disadvantage may be that the computer keyboard differs from the standard typewriter, although the differences are slight. It may matter little if one is only going to use these skills with a computer.

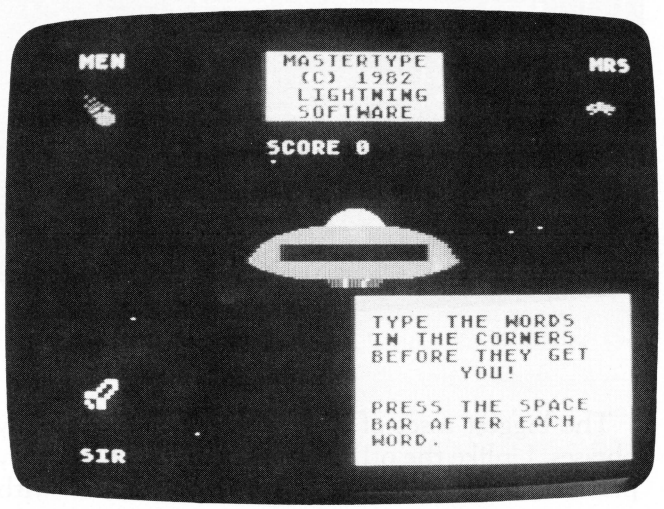
Some people type from copy, while others type as they compose. As a member of this latter group, I find the use of two fingers adequate to keep up with my sluggish mentation. For most people, however, these programs free one from thinking about fingers instead

of words and may be of great benefit. I heard that Isaac Asimov, maybe the most prolific writer ever, when asked what he would do if he only had one year to live, responded "Type faster!"

BONUS RIDDLE: What common 10-letter word can be typed out using only the top row letters (QWERTYUIOP)?

MASTERTYPE
Lightning Software
P.O. Box 11725
Palo Alto, CA 94306
32K/Disk Basic $39.95

As commander of a space ship located in the middle of the screen, the neophyte typist must destroy the combat enemy word stations appearing in each of the four corners, before their missiles destroy the ship. When a word or letter is correctly typed, a blast of energy emanates from the ship toward the respective word, and its missiles are exploded. All this fun and learning to type at the same time sounds too good to be true.



MasterType.

There are eighteen lessons, the first of which begins with the practice of single letters from the home row. The accompanying booklet includes a diagram indicating the proper positioning of the fingers on the home keys and the other keys for which each finger is responsible. As the lessons progress, the other row letters are incorporated in the drills, and multiple letter words are presented. Numbers, punctuation marks and shifted symbols are included in the more advanced lessons. After entering a word, the space bar must be pressed to fire the laser—a realistic approach, since this is what one would do if actually typing a word as part of a text. If a mistake is made, one may press the space bar and try again; however, the delete/backspace key is functional, and using it is better practice for the real world of typing.

Several options are available, including a level for beginners in which each word is only one letter long. The CHANGE mode allows one to vary the game speed, access a new lesson, or switch to upper/lower case letters. This latter feature is quite important and not found in all programs. Custom lessons may be created with forty words per lesson, with a maximum length of nine characters each. Sentences cannot be entered as such, because the use of the space bar terminates a specific entry. After each game, one's progress is charted by calculation of the typing speed in words per minute.

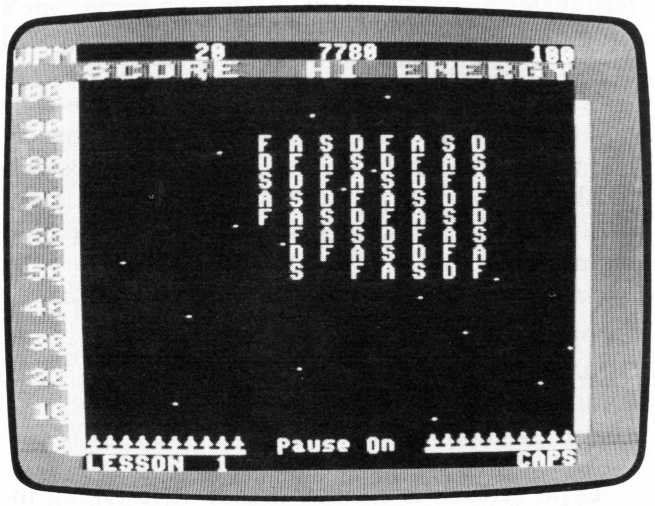
MasterType is a fast-paced, exciting, arcade-style typing trainer with clever sound and graphic enhancements. My children found this program the most fun to play. Its repetitive and progressive format make it a valuable tool for anyone wanting to develop typing skills.

TYPE ATTACK
Sirius Software, Inc.
Sacramento, CA 95827
16K Disk $39.95

Type Attack is another arcade-style typing tutorial, this time in the **Space Invaders** genre. Each lesson is composed of two parts. First is the character attack, consisting of three separate waves of eight columns of characters. As the bottommost character is typed, it is zapped from the screen, then the next, until the entire wave has been systematically destroyed. No aiming is required, but just like the game on which it is based, the columns gradually move toward one's bases bent on devastation. The character attack is followed by the word attack, comprised of groups of complete words using the previously practiced characters. The words fly horizontally across the screen, the one with the blinking shield being vulnerable to attack. Correctly entering the letters and pressing the space bar blasts the word from the sky. If not destroyed, the word wraps around the screen, giving one another opportunity. Incorrect entries at either level, or allowing a word to wrap around, consumes energy. When the energy runs out, the game is over.

A menu is displayed initially and offers several choices. The speed may be selected from settings of 1-99. (The manual suggests that robots and genetic mutants will feel most comfortable at speeds above 80, a fact I will not dispute.) There are thirty-nine preprogrammed lessons from which to choose. These progress in the order of traditional typing instructions practicing the home row first, etc. . . . Up to sixty additional lessons may be created and saved. Fifteen words with a maximum length of eight characters (without spaces) may comprise each lesson. The booklet contains diagrams for proper finger positioning for both the 400/800 and the XL series.

Typing speed in words per minute is displayed by a bar on the left side of the screen as the attack progresses. Points scored for letters destroyed and points computed from the speed level multiplied by the WPM result in a final score. The highest scores enter a "Hall of Fame." This program includes use of upper/lower case letters and backspace for entry errors. An additional feature is that a game in progress may be saved and resumed at a later time.



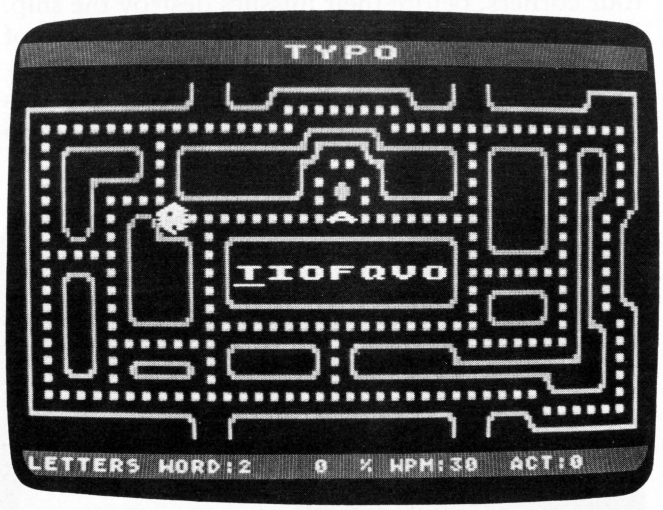
Type Attack.

The sound and graphics used in **Type Attack** make the overall presentation of this program quite enjoyable. Dexterity with the keyboard will definitely be achieved while having a lot of fun. This is the most addictive of the programs I have seen; and addictiveness leads to repetition, which is what typing instruction is all about. This program is my personal favorite, but just by a whisker. It is certainly well worth the investment.

TYPO
Romox Inc.
501 Vandell Way
Campbell, CA 95008
8K/Cartridge $39.95

So far, we have had a typing tutorial which blasts word enemies approaching from all sides — similar to many early arcade games — and one which annihilates menacing columns of characters a la **Space Invaders**. What's left? **Pac-Man**! In the middle of a dot-filled maze is a window where letters, words or phrases appear. As one types the contents of the window, a little ship wends its way through the maze-consuming dots. Of course, there is the ever present fuzzy monster chasing the ship through the maze. The monster moves at the rate of the preselected WPM (words per minute), requiring one to type faster than the WPM to avoid destruction. The object is to eat all of the dots while learning how to type.

While Beethoven's *Fur Elise* provides the background music, a number of options are displayed in the menu. The desired speed in WPM (1-120) is selected. One then has the option of practicing random letters, words or phrases. There is also the option to create one's own lesson. At the bottom of the playing screen can be seen the preset pace, as well as the actual typing speed in WPM. With the display window in the middle of the screen, it is often difficult — if not impossible — to follow the progress of the dot-ravaging ship and its pursuer, while trying to type the correct sequence of letters. There is no backspacing capability, since only the correct letter is accepted. An erroneous entry only delays the progress of the ship. Although shifted punctuation marks and symbols are permitted, there is no provision for upper/lower case characters, only capitals.



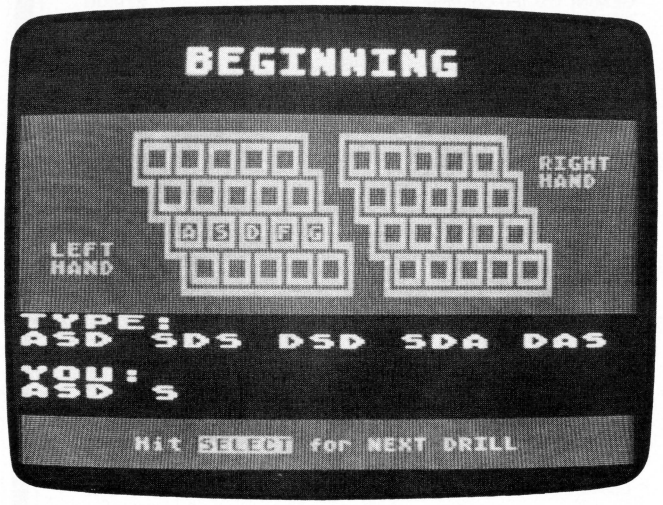
Typo.

The biggest asset of **Typo** is its ability to display phrases. Unlike the other two typing games, one may practice letter sequences, words or phrases with spaces, punctuations, etc. — with lengths of up to 114 characters. With this program, one can create exercises like *the quick brown fox* and *Now is the time for all good men*. The simplicity of a cartridge, plus pleasing sound and graphics, add to the learning process. **Typo** differs a little from the others, but it, too, is fun — and one's fingers will soon learn where to go.

TOUCH TYPING
ATARI
P.O. Box 427
Sunnyvale, CA 94086
16K/Tape Basic $24.95

Touch Typing provides a more traditional approach to acquiring typing skills. Progressing from Beginner to Advanced levels, repetitive drills are displayed on the screen for one to duplicate. Exercises with one hand, then the other, then both — all the

way to typing paragraphs — are found in this program. The manual is excellent, and the method of instruction is well founded. In the Beginner level, a keyboard is displayed on the screen, highlighting the letters being practiced. There is no upper/lower case provision at this level, although it is found at both the Intermediate and Advanced levels. Word and letter error counts, as well as typing speed (WPM), are kept and displayed at the bottom of the screen. Since this program incorporates the tracking of errors, there is no backspacing capability.



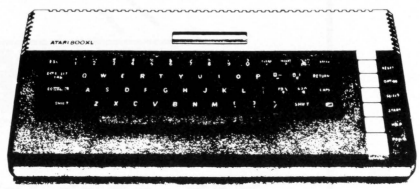
Touch Typing.

I have mixed feelings about the overall appeal of this program. It is less expensive, being on tape, but it seems to take forever to load. It is fundamentally sound and based on classic typing training methods, but it is somewhat boring compared to the other typing programs. Probably the most comprehensive of any of the programs, it may gather dust if the student is not highly motivated.

There you have it. Four different programs to teach typing skills. All are well conceived and serve their purpose. For those on a tight budget, a program to give one practice typing sentences appeared in **ANALOG** issue 6 and has been reproduced in **The ANALOG Compendium. Typing Trainer**, by Regena, is written in BASIC for 16K Tape/24K Disk and employs special effects in the form of a steam engine and whistle. The program can easily be changed to include a wide variety of sentences of one's own choosing.

Typing skills are important. Present day educators might say, "Maximal utilization of digital dexterity in interfacing with the computer will impact positively in the endeavor to forestall nonsuccess." Oh yes, the answer to the riddle which seeks the common 10-letter word using only the letters in the top row of the typewriter: "proprietor" and "repertoire" meet the requirements, but most apropos is, of course, *typewriter*. □

Always the Lowest Prices on the Finest Quality at. . . COMPUTER CREATIONS



ATARI PLATO SYSTEMS

The Electronic Highway To The Universe
Over 6000 Educational Programs

SYSTEM I	SYSTEM II
Atari 1030 Modem	Atari 850 Interface Module
Atari Plato Cartridge	Atari Plato Cartridge
	Anchor Mark XII Modem
$159	$549

ATARI HOME COMPUTERS

ATARI 600™ Home Computer (16K RAM) ..... CALL FOR
ATARI 800XL™ Home Computer (64K RAM) ..... UPDATED PRICES

DISK DRIVES

RANA 1000 .....
TRAK ATD2 (Single/Double Density/parallel int./4k buffer) .....
TRAK ATD4 (Dual Drive) ..... CALL FOR UPDATED PRICES
TRAK TURBO DOS .....
INDUS GT (Free Software) .....
ASTRA 1620 .....
ATR - 8000 .....

GENERIC DISKS

Generic 100% Defect-Free/Guaranteed MINI-FLOPPY DISKS
Diskettes (1 Box Min.) - 10 per box

	SS/DD	DD/DD
1 or 2 Boxes	17.49/box	20.99/box
3 - 9 Boxes	15.99/box	19.99/box
10+ Boxes	14.99/box	18.99/box

Bulk Diskettes with Sleeves - Price per Disk

	SS/DD	DD/DD
10 - 29	1.59	1.99
30 - 99	1.49	1.89
100+	1.45	1.79

HARDWARE COVERS

ATARI 400	5.95
ATARI 800	5.95
ATARI 1010	5.95
ATARI 410	5.95
ATARI 810	5.95
ATARI 1200	5.95
ATARI 1025	5.95
ATARI 600XL	5.95
ATARI 1050	5.95
ATARI 800XL	5.95
ATARI 1027	5.95
EPSON MX-80	5.95
EPSON FX-80	5.95
EPSON RX-80	5.95
EPSON RX-80 FT	5.95
EPSON FX-100	7.49
GEMINI 10X	5.95
GEMINI 15X	7.49
RADIX 10	5.95
RADIX 15	7.49
POWERTYPE	5.95
DELTA-15	7.49
RANA 1000	5.95
INDUS	5.95
TRAK	5.95

DISKETTE/CARTRIDGE/CASSETTE FILES

Flip 'N' File	$ 3.95
Flip 'N' File 15	6.95
Flip 'N' File 50	26.95
Flip 'N' File/The Original	19.95
Cassette 'N' Game File	14.95
Disk Bank	3.99
Disk Bank (Plastic)	1.99
Blank 30 Min. Cassettes	1.29
Blank 10 Min. Cassettes	.99

RAM (MEMORY) BOARDS

Intec 64K Board (400)	$109.00
Intec 48K Board (400)	89.00
Intec 32K Board (400)	39.00
Microbits 64K (600XL) Expansion	129.00

80 COLUMN BOARDS

BIT 3 ..... $249.00

PRINTERS

GEMINI 10X (80 Column)	279.00
GEMINI 15X (136 Column)	399.00
DELTA 10 (80 Column)	479.00
RADIX 10 (80 Column)	639.00
RADIX 15 (136 Column)	749.00
Powertype Daisywheel	379.00
Epson RX-80 (80 Column)	339.00
EPSON RX-80 (80 Column)	339.00
EPSON RX-80 FT (80 Column)	469.00
EPSON FX-80 (80 Column)	555.00
EPSON FX-100 (136 Column)	749.00
AXIOM AT-100 (80 Column) Includes Interface & Cable	229.00
SILVER-REED EXP 500 Daisywheel (80 Column)	399.00
OKIDATA 92P	449.00
TRANSTAR 315	549.00

PRINTER INTERFACE CABLES

APE FACE	65.00
ATARI Universal Cable	59.00
MPP-1150 Parallel Printer Interface	79.00
Parallel Printer Cable	30.00

PRINTER RIBBONS

Gemini Printers (Black/Blue/Red/Purple)	3.00
Epson Printers	6.00

MONITORS

Gorilla™ Hi Res 12" Non-Glare Screen	89.00
Gorilla™ Hi Res 12" Non-Glare Amber Screen	99.00
Sakata SC 100 Color Screen	239.00
NEC 1260	109.95
Monitor Cable	10.00

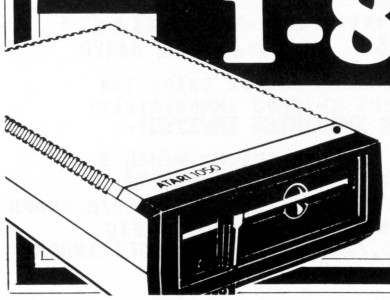
MODEMS

MPP-1000C Modern	129.00
Signalman Mark II Modem	79.00
Hayes Stack Smartmodem (300 BAUD)	239.00
Hayes Stack Smartmodem (1200 BAUD)	549.00
Novation J CAT	119.00
Novation 103 Smart CAT	189.00
Novation Auto CAT	199.00

ATARI ADDITIONAL EQUIPMENT

ATARI 850™ Interface Module .....
ATARI 1010™ Program Recorder .....
ATARI 1020™ 40-Column Color Printer/Plotter ..... CALL FOR UPDATED PRICES
ATARI 1025™ 80-Column Printer .....
ATARI 1027™ Letter Quality Printer .....
ATARI 1030™ Direct Connect Modern .....
ATARI 1050™ Disk Drive .....
CX77 ATARI Touch Tablet™ .....

To order call TOLL FREE
1-800-824-7506



COMPUTER CREATIONS, Inc.

P.O. Box 292467 - Dayton, Ohio 45429

For information call: (513) 294-2002 (Or to order in Ohio)

Master Card - Visa - C.O.D. (Add $2.50). All orders add $3.50 Shipping and Handling
Ohio Residents add 6% for Sales Tax. **ASK FOR OUR FREE CATALOG**

CIRCLE #114 ON READER SERVICE CARD.

BOULDER DASH
by Chris Gray and Peter Liepa
FIRST STAR SOFTWARE
22 East 41st St.
New York, NY 10017
32K Disk/Tape $29.95 16K Cartridge $39.95

by Tom Hudson

To be honest, the members of the **ANALOG** staff have never been very impressed with most of the software from First Star. Some of their programs, such as **Bristles**, had messy glitches; others just didn't cut the mustard in the fun department.

Needless to say, I was very pleasantly surprised when I saw **Boulder Dash**. In my opinion, this game tops all of First Star's previous efforts in terms of originality, long-lasting challenge, and just plain fun.

Original is best.

In **Boulder Dash**, you play the part of Rockford, a small quasi-human with the power to dig through the earth. Rockford's objective is to collect as many glittering diamonds as he can.

Rockford's job isn't that easy, though. He must dodge the boulders and diamonds which fall when he digs under them. One wrong step, and Rockford is only a fond memory!

Some levels contain Fireflies, flashing objects which kill Rockford on contact. Their behavior is predictable, so Rockford can avoid them, if he keeps his eyes open. Rockford can also kill the Fireflies by dropping boulders on them.

Butterflies are another danger for Rockford to look out for. They are similar to Fireflies, but move in the opposite direction. Unlike Fireflies, Butterflies turn into diamonds when they are killed.

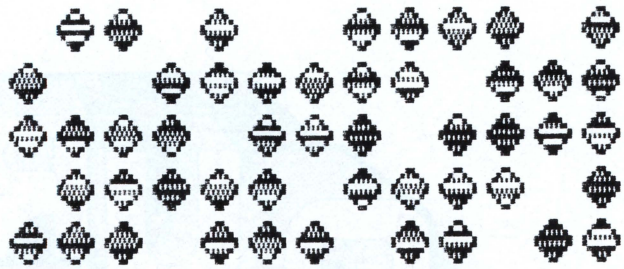
Other features, such as a growing green amoeba and enchanted walls (which turn boulders into diamonds and vice-versa) add to the challenge of this original game.

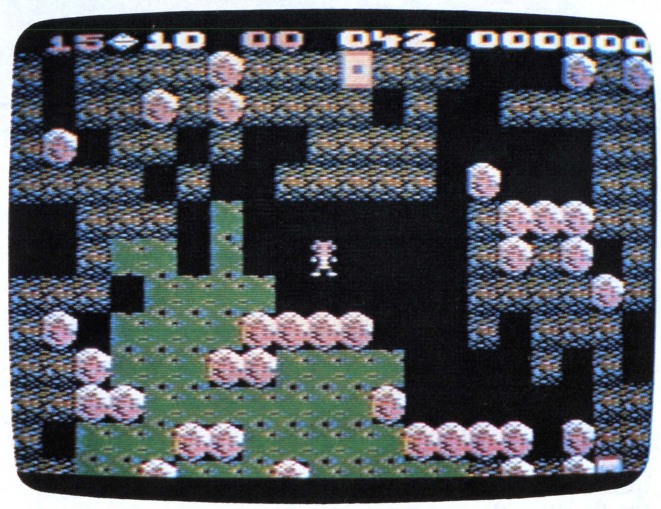
Long-lasting.

Each level of **Boulder Dash** takes place in a different "cave," made up of several scrolling screens. These caves range in difficulty from "extremely easy" to "almost impossible." There are sixteen caves in **Boulder Dash**, each with five difficulty levels and different puzzles to solve. Rockford's time in each cave is limited, so the screens must be completed as quickly as possible.

Unlike other games, **Boulder Dash** doesn't simply speed up the action on each level, but rearranges the obstacles and increases Rockford's quota of diamonds.

Every four caves, you are given the chance to solve a "playable intermission" screen. If you successfully complete the screen, you are awarded a bonus Rockford. Some of these intermission screens seem more difficult than the caves (even though I've made it through all sixteen caves, there is one intermission screen I can't get through — yet)!





Boulder Dash.

One nice thing about **Boulder Dash** is that all the caves are stored in memory while the game is being played. The computer doesn't have to access the disk each time a new level appears, so your disk is spared the extra wear.

Just plain fun.

Ever since **Boulder Dash** arrived, publisher Lee Pappas and I have been racing to solve each level. **Boulder Dash** is very addictive, because each new level presents new challenges.

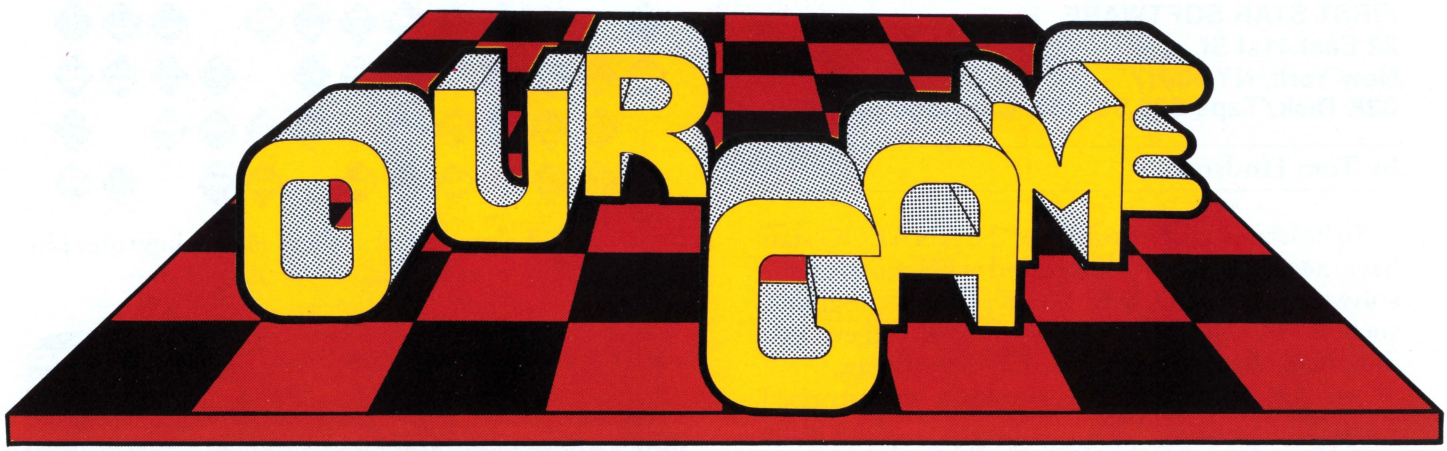
The level of detail in **Boulder Dash** is good, too. Whenever Rockford is standing still, his eyes will blink, he'll put his hands on his hips and tap his foot impatiently.

The bottom line.

Being a programmer myself, I couldn't help but have a small gripe about **Boulder Dash**.

While the figures of Rockford, the Fireflies and Butterflies are animated smoothly, they "jump" from one position on the screen to the next. When I first saw the game, this bothered me somewhat, but after playing the game for some time I have found that it does not affect the playability at all. **Boulder Dash** is, quite simply, an exceptional game, and one of the best yet for the Atari computer systems.

Incidentally, as further proof of this game's appeal, **Boulder Dash** has recently been adapted for arcade use by Exidy! □



by Joel Gluck

Welcome, once again, to **Our Game**, the only monthly column that brings you extra-large doses of both insight and insanity. Many readers have asked the question: "Joel, why do you seem so out of touch?" Well, I'll tell you: I write this column two months before it is published. For example, even though you are reading the July issue of **ANALOG** (which appears in June), I'm writing this column in April. Such a time delay can make things extremely difficult . . .

For example, there's the *Our Game Special Election-Year Game Idea Vote*, which I initiated last month (the June issue, which appeared in May). This is an election to determine which game idea you, the reader, most want to see developed in **Our Game** (see last issue for voting instructions and descriptions of the four nominees). All votes must be mailed by August 1st, 1984. The problem is, since I have towait until August for all the votes to be in, development of "our game" has to wait until an issue two months after that, due to the delay factor. So, "our game" can't be started until the October issue.

What this means is that I have four months of **Our Game** (July, August, September, and October) to do what I want! After careful consideration, I decided that the best thing to do would be to present a detailed treatise on the subtle relationship between ice cream consumption in Boston and the likelihood of global thermonuclear war. When I mentioned this idea to my closest friends, they laughed at me and began to abuse my priceless collection of eighteenth century floppy disks. I was ashamed and agreed that the only honorable thing to write about in the coming issues would be the development of a game . . .

Viewer what?

But before we proceed on to such trivialities as writing a game, let's take care of something really important: viewer mail! The amount of mail we're receiving these days is growing by leaps and, uh, bigger leaps, but that doesn't mean the quality is improving any . . . If you're going to write to **Our Game**, please stick to English (or, if you must, Mandarin Chinese or Swahili), and please use standard writing instruments — I'd like to remind a few of our readers that salad dressing just doesn't make very good ink. Please keep this in mind.

Our first letter this month comes from Matthew J.W. Ratcliff of St. Louis, Missouri. He has some useful additions to last month's tutorial on playtesting:

The less the playtester knows about your program, or programming in general, the better. This will immediately tell you several things, like how well you trap errors. A person who seldom computes will quickly find any major and, quite often, subtle bugs. If it's a utility program which requests a filename, for instance, what if the person types KEEP for a load file, instead of D:KEEP? Does the computer sit there, dumb as a rock, with no error codes? Does a CTRL-3 get out of the problem (keyboard-forced end of file)? What about other expected keyboard inputs; does CTRL-3 crash the program? Does the code disallow inverse video, shift clear, and the break key? I could go on and on about the trials and tribulations faced with expected keyboard inputs. Let a novice "fat-finger" the keyboard for a while, and you will find them quickly enough.

Thanks for the good tips, Matthew. Handling keyboard input is sometimes so frustrating that programmers often avoid it entirely, checking only for values from the joystick or the console keys

(START, SELECT and OPTION). Actually, for most games or educational programs, keyboard input can be an unnecessary complication — working off only the joystick, for example, can make a program more user-friendly.

Cecil C. Alton, of Dumfries, Virginia, writes:

I would like a game I could play with my two-year-old. He is fascinated with the computer and especially likes BASIC — where he keys in letters, both singly and with repeat feature. Other games I have interest him, and he grips the joystick with eager anticipation, but he does not interact with the game. One wild idea is to build a game with a simple "press any key" response being elicited from the player. This game would have to be easy to learn — i.e., no difficult instructions required — and could be developed around a learning-curve concept (learn from mistakes or from player's responses).

That's an excellent idea for a game, Cecil, and I believe someone's already done it! Monarch Data Systems, Inc., has just such a game — it is called **SofToy**. The game consists of nine cute little animated pictures on the screen, which are activated by pressing a key. In the easiest version, any key pressed will activate a picture. But the game can also be made harder, so that only certain numbers or letters will make the pictures move. At its toughest, **SofToy** presents a child with sequences which he or she must duplicate by hitting the correct keys, very similar to the electronic game called **Simon**. This program may be just what you're looking for, Cecil.

Tom Hull, of Wakefield, Rhode Island, has some strong feelings about **Our Game**, not to mention a very unusual game idea:

In my opinion, I think you're setting up too many rules. My dreams of sending you "dream game" ideas were shattered when I couldn't fit in simultaneous, multi-players and no violence, period! Some of the requirements seem ridiculous to me because of how easily they could be solved. So here are my words of wisdom on each of your requirements.

1. Violence: *I agree; the wrong type of violence could be harmful to the young minds of children. The "kill or be killed" theme should be avoided, but what about the "survive or be killed" theme? I don't see how saving your own skin would be harmful to kids' minds. Say the only objective is to run away from falling buildings or escape from a forest fire. How harmful can that be?*

2. Simultaneous play: *Once again, I agree. It is fun when you either play against or in cooperation with a friend, but what if none of your friends like the game? This is why one should have the option for either single or simultaneous play.*

3. Sex Difference: *This is the ridiculous one, because just a bit of good programming can solve the whole problem. Consider the following:*

```
10 ? "Do you want to be a boy or a gir
l";:INPUT A$
20 IF A$="BOY" THEN PLAYER$="!#$":REM
!#$ would be the character for a male
player when redefined.
30 IF A$="GIRL" THEN PLAYER$="@%&":REM
 @%& would be the character for a fema
le player when redefined.
```

This could be modified to use P/M graphics or whatever you'd want to use. Another method would be to use an animal to portray the player, say a turtle. That way, no one could accuse the turtle of being male or female, as long as you don't call the game **Mr. Turtle**, or **Turtle Man**.

Now that that's off my chest, let's get to the game idea, which I call **The Punkarium Wave**. *The setting is in a one-story mall. The player is an everyday person who just came out of the arcade and is about to go to the north end of the mall, where the person (you) has parked the car. Then you realize that, while you were in the arcade, the whole mall was taken over by punks, a class of people who all have mohawk haircuts, wear sunglasses and carry around "boxes" that are all blaring out the same punky tune (that sounds like someone trying to play a synthesizer like a bagpipe)!*

Their "lifestyles" are contagious, so you must avoid any contact with them — or you will become one of them! You run along a scrolling mall, trying to reach the north end, where the only remaining unlocked exit awaits. This would be impossible, if it wasn't for your only defense. Somewhere in the mall is Marvin's House of Metal. If you can find it and get inside, you can turn on the mall's speakers and blare some heavy metal to drown out the punks' boxes. All of the punks will stop dead in their tracks and cover their ears, letting you skip on by them. In ten seconds, the punks will have turned off the speakers. If you are not out of the mall by then, the punks will rush to block the north exit and all hope will be lost.

Well, Tom, I think your **Punkarium Wave** wins **Our Game**'s "Weird Idea of the Month" award (your prize, a peanut butter and avocado sandwich, is in the mail). As for your complaint about **Our Game**, having "too many rules," let me say that there are no "rules" as to what you can *send* to **Our Game**. I like to see all kinds of game ideas, whether they be violent or non-violent, one or two-player, or whatever. The reason I've expressed a preference for non-violent games is simply that there have been so many violent video games that I am rather bored with the concept. It takes creativity and imagination to come up with something really new, and it is my challenge to the readers to submit non-violent games. It doesn't mean they have to.

As for your quick solution to the question of games that are biased toward one sex, I'm not so sure that changing the graphics is all that is needed. I believe that the general subject matter of most video/computer games tends to attract males more than females. Again, it's a challenge to the readers to come up with something different.

The task is not impossible. Last month, to kick off *Our Game Special Election-Year Game Idea Vote*, I nominated four game ideas, all of which were based on reader input, and all of which were essentially non-violent, two- or multi-player, and none of which seemed sexually biased (except maybe for Idea #1, which has a husband and his "huge wife," but that can be modified).

Our last letter this month comes from Greg Rizzo of Chicago, Illinois:

The truck that delivers peanuts to the zoo is late. You, the elephant, become very hungry. When the truck finally arrives, it is in such a hurry that it crashes and spills peanuts all over the zoo. You become so hungry that you break out of your cage and travel all around the zoo, shown on the TV screen as a maze, looking for and eating peanuts. But be careful, because there are mice wandering around the zoo. If they touch you, they will scare you to death. Also, there is a zookeeper who will appear on the screen looking for you. But, for your protection, there are mousetraps set at random spots in the zoo. You get points for eating peanuts and for catching mice in mousetraps. But you will lose a life for getting scared to death by a mouse.

I must admit it wasn't really my idea. It was really my brother's and his friend's. I just expanded on the idea.

Greg! How could you? Stealing your brother's game idea like that! Tsk, tsk. It's a nice game idea (I like the story behind the game, especially), but the game play itself sounds suspiciously like **Pac-Man**. What if, instead of being the elephant, you were the zookeeper? The elephant is loose in the zoo, eating spilled peanuts. Your aim is to get the elephant back into his cage as fast as possible. You do this by closing and opening gates in the zoo/maze, and by moving many of the peanuts so that they make a trail leading back to the elephant cage. To make the game more interesting, the maze could be different every time.

Well, that's it for viewer mail this month. Even though the *Our Game Special Election-Year Game Idea Vote* is in progress, don't hesitate to send in any new idea you have. If it's any good, it'll probably appear in these pages — which means that people all over the U.S.A., not to mention the entire world, will see *your* name and read *your* idea!

Clues.

This month, and the next three months of **Our Game**, will be devoted to a discussion of the creation and development of a simple computer game.

The working name for this game is **Clues**, and the first prototype version, CLUES.A, appears in Listing 1.

The idea behind **Clues** is very simple, and not entirely new. When playing, you are presented with a grid underneath which there is a buried treasure. To find the treasure, you move your man (whom I call the Seeker) to a likely spot and hit the trigger. If you were correct, you win. If not, the computer gives you a clue as to where the treasure is.

The clue is either an arrow or a number. An arrow points in the general direction of the treasure. A number gives the approximate distance of the treasure from your current spot.

This is not a new idea. I believe that there was a game of this type for the Atari 2600 (way back when it was called the Video Computer System). In that game, you were looking for a flag, not a treasure. Big difference . . .!

Of course, the CLUES.A is a simple one-player game. More later about how we can improve and expand it.

Explanations.

Unlike the dreaded FLW listing from issue 16, Listing 1 is fairly clear. There is no mysterious string manipulation or brain-damaged program logic, and everything is simple and well documented with plenty of REMarks. Note: *When typing the listing in, do not omit REM's* that appear alone on a line. These are frequently accessed by GOTO's and GOSUB's.

And now, an **Our Game** first . . . a detailed explanation of the program:

Lines 200-260 are the top level of the program. The way it is organized, into five GOSUB's (with REMarks), makes the program very easy to read and follow, and makes finding specific parts of the program simple (for example, if you want to change something in the screen setup, Line 220 informs you that the screen initialization code begins at Line 3000). I usually begin all large BASIC programs with a series of GOSUBs like this.

Notice that Line 250 assumes that a variable called PLAYAGAIN was given a value at some point, probably in the subroutine starting at Line 5000. If PLAYAGAIN=1 (1 meaning "yes" or "true"), then the game branches back to the screen initialization routine.

Getting Ready.

The routine starting at Line 1000 prints the instructions and waits for the user to press the START key (Line 1200 handles that). The subroutine is called "Intro/Options," because if there were any game options they would appear at this point.

Starting at Line 2000 is the initialization procedure. Lines 2100-2250 handle the joystick data. The problem with the Atari joystick is this: what you'd like to have is the horizontal and vertical direction of the joystick (indicated by -1, 0, or 1 for each. For example, a vertical direction of -1 means "up," and a horizontal direction of 1 means "right." Zero means there is no movement along that component), but what the joystick gives you is a value from 5 to 15 that stands for one of the eight directions. To convert from this value to the horizontal (X) and vertical (Y), I READ -1's, 0's, and 1's into 2 arrays (XS() and YS( )) indexed off the joystick value. For

WHAT IS D:CHECK/C:CHECK?

Most program listings in **ANALOG** are followed by a table of numbers appearing as DATA statements, called "CHECKSUM DATA." These numbers are to be used in conjunction with D:CHECK and C:CHECK, which appeared in the **ANALOG Compendium** and Issue No. 16.

D:CHECK and C:CHECK are programs by Istvan Mohos and Tom Hudson. They are designed to find and correct typing errors when entering programs from the magazine. For those readers who do not have a copy of either article, send for a copy of back issue 16 ($4.00) or **The ANALOG Compendium** ($14.95 plus $2.00 shipping and handling) from:

ANALOG Computing
P.O. BOX 615
HOLMES, PA 19043

example, if the joystick reads "6" (up and to the right), the value given by YS(6) is -1 (up) and the value given by XS(6) is 1 (right).

The different characters used for "arrows" in the game are in DATA on Line 2360 (in future versions of the game, we'll redefine the character set to have better-looking arrows). The ASCII codes of these are read into the array ARROW( ) in the loop starting on Line 2320. Notice that I have to READ each arrow using the small string called CH$, before storing the ASCII value of CH$ (plus 128 to make it reverse field — the "negative" image of the character) into the ARROW( ) array.

The ASCII codes for other characters that will appear on the screen are stored in aptly named variables starting on Line 2400. The GRID character, for example, is a period (.) and the SEEKER character is the solid ball graphic (CTRL-T). These characters, too, will be modified in future versions.

Screen initialization begins on Line 3000. The game itself is in graphics zero, the normal text mode, so, to make it look a little different, the screen and border colors are changed on Line 3110. Line 3120 uses a nifty POKE 752,1 which hides the cursor.

Starting on Line 3200, we see something interesting: COLOR WALL. Now we know that the variable WALL was defined as the ASCII code of a reverse field space (a solid white block) on Line 2410. We also know that COLOR is ordinarily used in plotting modes like 3, 5, and 7 to select a color register to draw with. Well, it so happens that invoking COLOR with the ASCII code in a character mode lets you draw with that character using PLOT's and DRAWTO's. This is exactly what happens on Line 3210, which draws a wall using the WALL character around the screen.

Lines 3250-3280 use a similar technique to draw the grid. COLOR GRID selects the appropriate character, and the loop does the rest. Lines 3300-3310 set up the starting coordinates of the Seeker (the approximate middle of the screen) and plot it. There is also a variable called UNDER, to store the value of what is under the Seeker (initially, plain old grid character), in case the player moves the Seeker over some of the clues he has dug up.

Lines 3400-3420 set up the treasure, and make sure its position is not equal to the Seeker's starting position.

The game.

The operating code for the game itself begins on Line 4000. Right before it begins, the timer is set to zero on Line 4100 (the Atari has a real-time clock measured in sixtieths of seconds — jiffies — at memory locations 18, 19, and 20), and the number of GUESSES is set to zero at Line 4110. This is so we can tell the player how long and how many guesses he or she took to find the treasure when the game is over.

Lines 4200-4240 are the nucleus of the game. All actions stem from these lines. The stick and trigger values are stored. If the trigger is being pressed and the stick is still (Line 4220), it means the player wants to venture a guess, so the program branches to the "take a guess" subroutine. If the joystick isn't idle (Line 4230), then the Seeker must be moved, so the program branches to 4300. If neither of these conditions are met, then the program does nothing and loops back to get new values for the joystick and trigger.

The routine for moving the Seeker (starting on Line 4300) contains a POKE 77,0. This is to prevent the computer from going into "attract mode" (color flipping), which occurs if the keyboard isn't used for about nine minutes. This poke is in the movement routine, so that if the player has stopped playing the game, the poke won't be executed, and after nine minutes the computer *will* go into attract mode.

Line 4310 uses the joystick direction arrays we created (you remember, way back in the initialization routine!) to convert the joystick value (S) to horizontal direction (XD) and vertical direction (YD). Line 4320 looks one spot ahead of the Seeker in the current direction, and stores the ASCII value of what's there into the variable G (that's how the LOCATE command works — consult your BASIC Reference Manual for details). If G is equal to the value of WALL (Line 4330), that means there is wall ahead of the Seeker. The Seeker isn't supposed to move through walls, so the program goes back to the game loop.

To move the Seeker, we erase it, update its position, and redraw it. This happens quite clearly on Lines 4350 to 4370. The only trick is, instead of erasing the Seeker with a blank space, we are erasing it with what's underneath it (Line 4350), whether it be a grid or an old clue. Then, on Line 4380, the variable UNDER is given the value of G, which is what's under the Seeker now.

The "take a guess' routine, starting on Line 4500, is a bit more complex. First, it increments the number of guesses (Line 4502) and then proceeds along the following logic:

Line 4505 — If the guess is correct (a win), pop out and return to the top level.

Line 4520 — If what's under the Seeker is an old clue, jump ahead (to 4700) and display that clue.

Line 4530 — Compute the distance from the treasure.

Line 4540 — If the Seeker is too far away to give a one-digit distance clue (or if a random whim is heeded), jump ahead (to 4600) to get an arrow clue.

(***Our Game*** *continues on page 88)*

ANNOUNCING A MAJOR BREAKTHROUGH FOR ATARI COMPUTERS:

The Superbly Engineered Indus GT™ Disk Drive *The Critically Acclaimed Miles Accounting System II™*

PLUG-IN BUSINESS/ ACCOUNTING POWER.

These state-of-the-art components
transform *any* Atari computer
into a powerful, productive business system.

Here's the Facts on just how

Go from 88K (or 127K) to an Astounding 180K!

Your Indus GT disk drive opens the door to a whole new world of previously untapped computing power. The fact is, the other parts of your Atari, microprocessor and memory, are identical to those of an Apple //e. All your Atari ever needed was more disk storage... and with Indus you've *got* it.

EQUIVALENT SYSTEM PRICE COMPARISONS:

	ATARI 800XL	Apple //e	IBM PC
Computer w/64K and 2 Disk Drives	1297	2445	2633
Monitor with Interface and Cable	121	incl	680
Printer Interface and Cable	134	120	205
Printer	449	449	595
TOTAL HARDWARE	**1991**	**3014**	**4113**
General Ledger	145	395	595
Accounts Receivable	145	395	595
Accounts Payable	145	395	595
Payroll	FREE!	395	595
TOTAL SOFTWARE	**435**	**1590**	**2380**
TOTAL PACKAGE	**2426**	**4594**	**6493**

Based on Manufactuers Suggested Retail Price as of 4/84. Actual dealer price may vary.

Compare Indus GT performance with other drives.

Housed in a sleek, compact cabinet, the Indus GT coordinates perfectly with your Atari computer.

Reach track 39 in *less than one second*. Increase your Atari's transfer rate *400 percent* using the exclusive Indus SychroMesh DataTransfer™ mode. Compare that to others' drives.

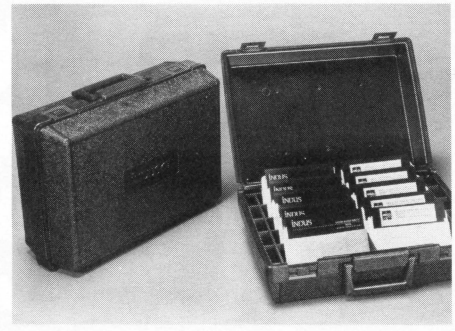
And while you're comparing, consider the built-in software that comes with every Indus GT: A comprehensive word processing package, an electronic spreadsheet, and a data base manager.

That adds up to formidable Atari performance at about *half the price of an Apple*; hardly a "games" machine!

Reliability. Convenience. Protection. Take them for granted with Indus.

A smart, smoked-glass dust cover, activated by air-drive. The AccuTouch™ push-button control system. These are just two of the *extra* operating innovations built into your Indus GT. These big steps beyond the ordinary assure hassle-free daily operation, and unusually long drive life.

Innovations in convenience abound. Automatic display of head-positioning; LED-lit *Busy* and *Protect* lights, and a two-digit LED register, only scratch the surface. And *every* Indus GT includes a rugged, molded PortaCase™ that doubles as an 80 disk library.



The Indus GT is also the *perfect* storage peripheral to power the incomparable, Miles Accounting System II.

Please send me information on Indus GT disk drives and the Miles Accounting System II business software.

YOUR NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

YES NO

☐ ☐ I currently own an Atari. Model # _____

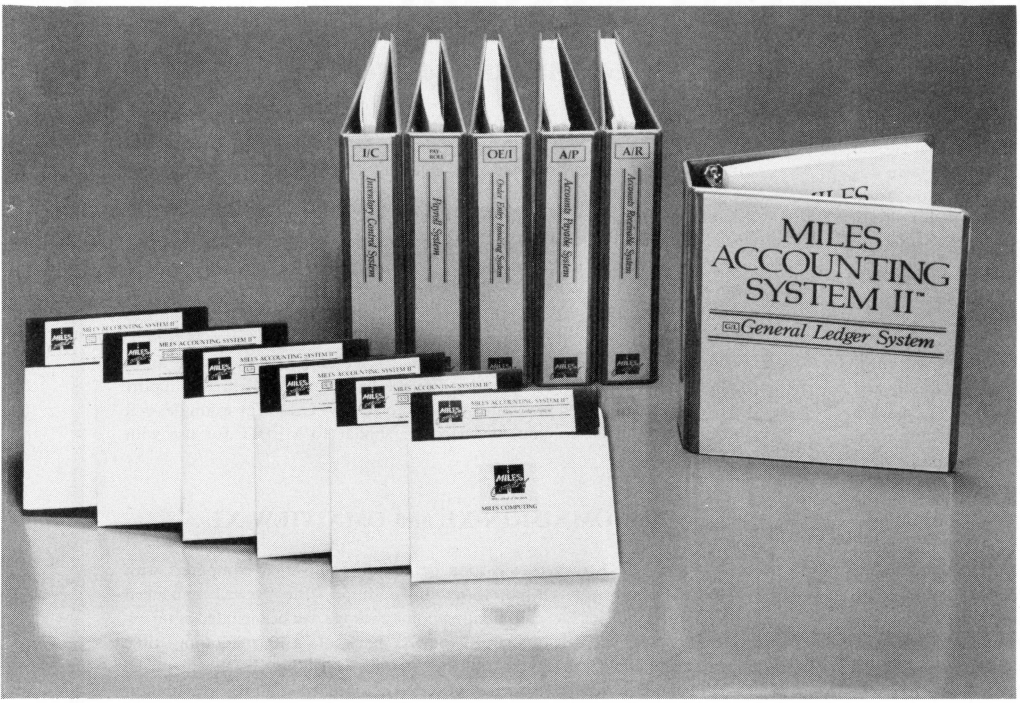
☐ ☐ I currently own a disk drive. Brand: _____

☐ ☐ I currently own Miles Accounting System II modules. They are: _____

INDUS

INDUS SYSTEMS
9304 Deering Avenue
Chatsworth, CA 91311
(818) 882-9600

effective your Atari can be:



Take absolute, effortless Command of your firm's Accounting needs...

With one swift stroke.

There are six modules in the Miles Accounting System II. Each can function independently of the others. But when linked together they can transform your Atari from a mild-mannered games machine into a powerful, integrated business productivity tool.

Powerful but Simple.

For all of it's versatility and sheer power, your "new" Atari system is utterly simple to use... even for beginners.

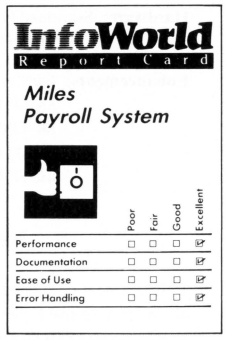
As ingenious as the Miles Accounting System II is, the true genious is that the programs have been written to enable virtually *anyone* to master them quickly. And the documentation is excellent; logical, clear and concise.

User support is guaranteed because Miles Computing is a company that *cares* about its customers. That is evidenced by their willingness to show you before you buy. With that in mind, drop by your local computer store and see our self-running demonstration of the Miles Accounting System II.

The Finest Accounting System available for the Atari today.

The critics agree. Miles Accounting System II is not only the finest accounting system available for the Atari today, but it rivals any system available for other personal computers. And it's the other half of the astonishing new Atari business power package.

While part of the overall Accounting System, the Miles Payroll System is particularly esteemed. Here's what respected computer industry publications have to say about it:



"In all respects—features, performance, ease of use, documentation and support—this is a professionally conceived and executed program. We look forward to other business software from Miles Computing that will serve the needs of Atari owners."

"The performance of Miles Payroll system is excellent. It handles all its promised features quickly and easily. ...It is an easy-to-learn program that is very powerful."
—InfoWorld

"So much for the Atari's reputation of being a games-only machine."
—Desktop Computing

MILES Computing

MILES COMPUTING INC.
7136 Haskell Avenue, Suite 300
Van Nuys, CA 91406
(818) 994-7901

CIRCLE #117 ON READER SERVICE CARD.

Here's the Bottom Line:

Buying an Atari computer is the best move you can make. With an *Indus GT* and *Miles Accounting System II*, there aren't any computer systems that can equal its new price/performance.

Special Offer.

If you act quickly, you can select any three Miles Accounting System II modules, and get a fourth of your choice **absolutely FREE**.

Call today for more information, or mail the coupon in this ad.

Apple //e is a trademark of Apple Computer Inc. IBM PC is a trademark of International Business Machines Inc.

The Latest Innovations From CDY For Your Atari System

OMNIMON! Resident Monitor

ANTIC July '83 review by David Duberman:
"OMNIMON! by David Young is a machine-language monitor that should have come with the ATARI. In fact, every microcomputer should have this sort of hardware based monitor installed. Most, however, do not. Now, for a relatively low cost, you can equip your ATARI 400/800 with a truly sophisticated programming tool. Whether you're an experienced programmer or a wondering beginner, OMNIMON can, if wisely used, help you to fully understand the working of your computer."

ANALOG July '83 review by Brian Moriarty:
"OMNIMON! can be a great addition to your ATARI computer if you know what to do with it. The ability to "freeze" a running program on-the-fly and examine the hardware registers is invaluable for testing and debugging; the sector-level disk functions are alone worth the price of the board . . . OMNIMON! might be one of the smartest investments you can make."

September '83: "Those of you who read my review of OMNIMON! in issue #12 know what a godsend it is for serious programmers. This ROM-resident monitor has saved me many hours of program development and debugging time, and recently made it possible for me to recover several otherwise unsalvageable text files that were lost when my word processor accidentally destroyed a disk directory. Ironically, the review you are reading is one of those salvaged files! Three of the ATARIs in our offices are now equipped with OMNIMON! boards, and more are on the way. Staff programmers Tom Hudson and Charlie Bachand both swear by OMNIMON!"

What is OMNIMON!?

OMNIMON! is a PC board which plugs into your 400/800 (soon to be available for the XLs also) and gives you complete control of your computer. Even though it is always available (by pressing SELECT and SYSTEM RESET) it takes up no user memory because it resides in the unused 4K block at $C000. Use it to interrupt, examine, and manipulate any program in memory whether it be disk, cassette, or cartridge based. It is especially good for program development or customization of existing programs. The flexible disk I/O allows you to write to or read from disk in either single or double density. You can edit raw sector data or even load a file without DOS. Many debugging tools are at your disposal: Display/Alter memory or 6502 registers, Disassemble memory, Search memory, Hex/Char modes, Single Step execution, JSR or GOTO address, Push/Pull stack, Printer dump, etc. After interrupting a program with OMNIMON!, many times it is possible to return to the program as if you had never left it (e.g., BASIC, DOS, etc.). Instructions are provided for the addition of a simple toggle switch to make OMNIMON! invisible, thus making it compatible with all software. An external cable is now provided to eliminate the need to solder directly on the board.

New 8K OMNIMON! Upgrade

This enhancement, which is available to all OMNIMON! users, includes a substantial number of features not available in the standard version. The 8K OMNI resides in an 8K ROM which has been modified by the addition of a switch for selecting either of two 4K banks. The additional features include Hex Conversion and Hex Arithmetic, Block Move, a Relocater, and a Line Assembler. A Binary Load command allows you to load any binary load file without DOS and doubles as a disk directory command which prints out the start sector of each file. Lockup recovery allows you to recover from system lockup, meaning that when your computer freezes, you can usually salvage the program or text file in memory by popping into 8K OMNI and dumping memory to disk. Advanced users will like the user extendibility feature which allows them to make use of the interface routines of 8K OMNI in their own software. One of the most exciting features of the 8K OMNI is the resident Ramdisk handlers. They allow AXLON Ramdisk owners to use this powerful device with any DOS which uses standard SIO calls and even with boot programs like word processors and games which access the disk a lot. Several additional features make this version very valuable for advanced programmers, but if you have a Ramdisk, 8K OMNI is a MUST!

New OMNIVIEW 80 Column Upgrade

Did you know that for most applications you do not need an expensive, slot consuming 80 column board to enjoy the power of 80 columns? Would you 400 owners like the convenience of 80 columns? OMNIVIEW takes advantage of the high resolution graphics mode built into the ATARI to generate an 80 column screen editor essentially identical to the ATARI screen editor (E:, S:). Thus, you can use OMNIVIEW in any environment where you would normally use the 40 column "E:" (e.g., BASIC, Assembler/Editor, etc.). The 80 column "E:" of OMNIVIEW has been optimized for speed so that it is not significantly slower than 40 column "E:". In addition, the character font was specially designed to be legible on an ordinary TV set! A monitor is recommended, but not really necessary for casual 80 column operation. The Bit-3 version of LJK's 80 column Letter Perfect has been modified to support OMNIVIEW and other programs are sure to follow. CDY, for example, will soon publish an 80 column screen editor similar to MEDIT for use with OMNIVIEW.

New OMNIMON-XL and OMNIVIEW-XL

600XL and 800XL owners will soon be able to equip their computers with OMNIMON and/or OMNIVIEW. In addition, the Newell enhanced operating system and Fastchip floating point package will be included at no extra charge. This will essentially turn your XL back into a 400/800 compatible machine and allow it to run most of the software which the XL-OS will not. A switch will allow you to select the XL-OS when needed. Call for availability.

Pricing

Hardware:	Standard OMNIMON! Piggyback Board	$99.95
	OMNIMON-XL/OMNIVIEW-XL	CALL
Enhancements:	(subtract $5.00 if ordered with board)	
	8K OMNIMON Enhancement	$45.00
	8K OMNIVIEW Enhancement - (4K OMNIMON with 4K OMNIVIEW)	$45.00
	4K OMNIVIEW Enhancement	$30.00

Newell RAMROD OS Board

This is a new operating system board which replaces the existing OS board. It allows you to use EPROMs in place of the ATARI OS ROMs and comes with an enhanced OS which includes additional graphics modes and a fast cursor. It also has a socket which will accept any version of OMNIMON and thus is an alternative to the OMNIMON! piggyback board. For the 800 only.

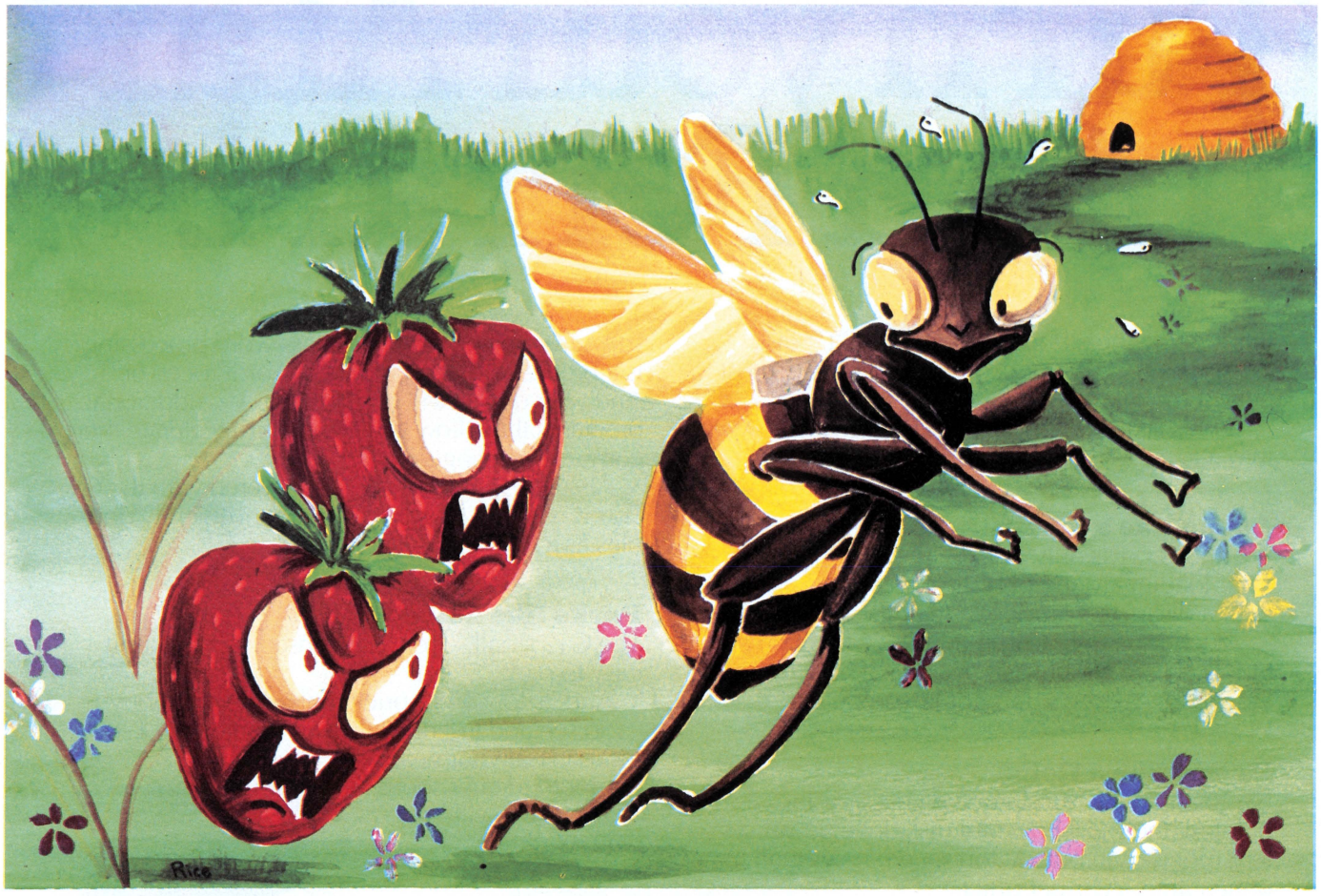
RAMROD OS Board with Standard OMNIMON	$149.95
RAMROD OS Board with 8K OMNIMON or 8K OMNIVIEW	$189.95
Same as above with Fastchip Floating Point Package	$209.95
Fastchip Floating Point Package by itself	$29.95

How To Order

We accept checks, money orders, or credit cards (Visa or MC). COD must be payed with cash or MO. Our toll free order desk is (800) 227-3800 ext. 561. Call this number only if you know **exactly** what you want and be explicit! They can answer no questions. For faster service or if you have questions please call or write:

CDY Consulting
421 Hanbee
Richardson, TX 75080
(214) 235-2146

DEALER INQUIRIES SOLICITED
CIRCLE #118 ON READER SERVICE CARD.



BUZZ-ZAP

16K Cassette or Disk

by David Karp

In the game of **Buzz-zap!** you are Stanley the Bug on his way to work. However, this is not an ordinary morning. This morning Stanley is pursued by a pair of killer strawberries and is trapped in a maze of deadly no-pest strips. As if this isn't bad enough, the hive he works in seems to be (and is!) moving away from the hapless bug.

Buzz-zap! is written in Atari BASIC with two machine language subroutines, called with the USR command. The first of these is Tom Hudson's P/M mover subroutine (issue 10, page 73), the second is just to flash the title screen. In the program's main loop first the stick is read, then Stanley is moved accordingly. Then the strawberries are moved so that they go towards Stanley. Then the hive is moved away from Stanley. Lastly, the collision registers are checked to see if Stanley has touched the walls, the strips, the berries or the hive. One point of interest is that each time Stanley gets to work (each board), the variable B is incremented and POKEd into location 201 decimal for storage until the title screen prints it. This way the score or number of boards is recorded.

Turn your Atari into a Ferrari.

Introducing the all-new 1984 Indus GT™ disk drive. The most advanced, most complete, most handsome disk drive in the world.

A flick of its "Power" switch can turn your Atari into a Ferrari.

Looks like a Ferrari.
The Indus GT is only 2.65" high. But under its front-loading front end is slimline engineering with a distinctive European-Gran flair.

Touch its LED-lit CommandPost™ function control AccuTouch™ buttons. Marvel at how responsive it makes every Atari home computer.

Drives like a Rolls.
Nestled into its soundproofed chassis is the quietest and most powerful disk drive power system money can buy. At top speed, it's virtually unhearable. Whisper quiet.

Flat out, the GT will drive your Atari track-to-track 0-39 in less than one second. And when you shift into SynchroMesh DataTransfer,™ you'll increase your Atari's baud rate an incredible 400%. (Faster than any other Atari system drive.)

And, included as standard equipment, each comes with the exclusive GT DrivingSystem™ of software programs. World-class word processing is a breeze with the GT Estate WordProcessor.™ And your dealer will describe the two additional programs that allow GT owners to accelerate their computer driving skills.

Also, the 1984 Indus GT is covered with the GT PortaCase.™ A stylish case that conveniently doubles as a 80-disk storage file.

Parks like a Beetle.
The GT's small, sleek, condensed size makes it easy to park.

And its low $449 price makes it easy to buy.

So see and test drive the incredible new 1984 Indus GT at your nearest computer dealer soon.

The drive will be well worth it.

INDUS™

The all-new 1984 Indus GT Disk Drive.

The most advanced, most handsome disk drive in the world.

For additional information, call 1-800-33-INDUS. In California, call 1-800-54-INDUS. 213/882-9600.

CIRCLE #119 ON READER SERVICE CARD.

Line documentation.

Lines 10-140 — Initialize; colors, shapes, M/L routines.

Lines 150-450 — Main loop; read stick, move bug, move hive, move berries, check collisions.

Lines 460-480 — Death routine, re-initialize.

Lines 490-510 — Next board routine, end.

Lines 520-770 — Title screen.

Lines 780-840 — Draw board routine.

Lines 850-870 — Tom Hudson's P/M mover initialize.

Lines 880-890 — Data for title screen M/L.

Lines 950-980 — Data for P/M mover M/L.

Lines 950-980 — Player shape data.

Lines 990-2060 — Board data.

Variables.

C,I,N,Q,Z — Working, data, or loop variables.
A — USR variable.
B — Board flag.
MOVE,PMMOV$ — M/L variables for P/M mover.
PMBASE,PMB — P/M base variables.
P$,PL$,PS$ — Player shape data strings.
P,PL,PS — ADRess of above.
X,Y — Stanley's position.
EX, EY, GX, GY — Berries' position.
FX,FY — Hive's position.
AA — Stick variable.
XI,YI — Modifiers to X and Y.
X1,Y1,X2,Y2 — Coordinates of strips.

```
10 POKE 201,0
20 FOR C=0 TO 3:SOUND C,0,0,0:NEXT C
30 GOSUB 610
40 FOR C=0 TO 3:SOUND C,0,0,0:NEXT C
50 GOSUB 850
60 DIM P$(5),PL$(6),PS$(6):P=ADR(P$):P
L=ADR(PL$):PS=ADR(PS$)
70 FOR I=1 TO 5:READ N:P$(I)=CHR$(N):N
EXT I
80 FOR I=1 TO 6:READ N:PS$(I)=CHR$(N):
NEXT I
90 FOR I=1 TO 6:READ N:PL$(I)=CHR$(N):
NEXT I
100 GRAPHICS 3:POKE 559,46:COLOR 2:PLO
T 0,0:DRAWTO 39,0:DRAWTO 39,19:DRAWTO
0,19:DRAWTO 0,0:POKE 752,1
110 POKE 16,64:POKE 53774,64
120 POKE 704,15:POKE 705,66:POKE 706,2
55:POKE 707,68:POKE 53278,244:POKE 623
,1
130 GOSUB 780
140 X=55:Y=86:EX=53:EY=19:FX=181:FY=33
:GX=197:GY=88:POKE 53278,244:SOUND 0,2
55,14,4:SOUND 1,233,14,2
150 FOR Q=1 TO 3
160 AA=STICK(0)
170 XI=(AA=7)-(AA=11):YI=(AA=13)-(AA=1
4)
180 XI=XI+(AA=6)+(AA=5)-(AA=10)-(AA=9)
:YI=YI+(AA=9)+(AA=5)-(AA=10)-(AA=6)
190 X=X+XI:Y=Y+YI:POKE 53278,244
200 A=USR(MOVE,0,PMB,PL,X,Y,6)
210 NEXT Q
220 POKE 53278,244
230 IF PEEK(53252)=2 THEN 460
240 EX=EX-(EX>X)+(EX<X)
250 EY=EY-(EY>Y)+(EY<Y)
260 A=USR(MOVE,1,PMB,P,EX,EY,5)
270 POKE 53278,244
280 FX=FX+(FX>X)-(FX<X)
290 FY=FY+(FY>Y)-(FY<Y)
300 A=USR(MOVE,2,PMB,P5,FX,FY,6)
310 POKE 53278,244
320 IF FX<55 THEN FX=55
330 IF FX>193 THEN FX=193
340 IF FY<25 THEN FY=25
350 IF FY>82 THEN FY=82
360 GX=GX-(GX>X)+(GX<X)
370 GY=GY-(GY>Y)+(GY<Y)
380 A=USR(MOVE,3,PMB,P,GX,GY,5)
390 POKE 53278,244
400 IF PEEK(53252)=2 THEN 460
410 IF PEEK(53260)=4 THEN 490
420 IF PEEK(53260)=2 THEN 460
430 IF PEEK(53260)=8 THEN 460
440 POKE 53278,244
450 GOTO 150
460 REM *****
470 POKE 53277,0:GOSUB 570
480 POKE 201,B:CLR :GOTO 20
490 REM *****
500 POKE 53278,244:GOSUB 520:GOTO 100
510 END
520 FOR I=250 TO 6 STEP -2
530 SOUND 0,I,10,10
540 FOR Z=1 TO 3:NEXT Z
550 NEXT I
560 RETURN
570 FOR I=255 TO 200 STEP -1
580 SOUND 0,I,10,10:SOUND 1,I-20,10,10
590 NEXT I
600 RETURN
610 FOR I=1664 TO 1692:READ N:POKE I,N
:NEXT I
620 GRAPHICS 18:SOUND 0,128,10,10
630 POKE 53277,0:POKE 16,64:POKE 53774
,64
640 ? #6:? #6:? #6;"      BUZZZ-ZAP!"
650 A=USR(1664):SETCOLOR 0,0,13:SOUND
1,192,10,7
660 ? #6:? #6;"    BY DAVID KARP"
670 A=USR(1664):SOUND 3,240,10,7
680 ? #6:? #6;"     PRESS START"
690 FOR C=1 TO 3:SOUND C,0,0,0:NEXT C
700 SOUND 0,128,10,14:FOR I=1 TO 30:IF
 PEEK(53279)=6 THEN RETURN
710 SOUND 1,192,10,12:NEXT I
720 SOUND 0,224,10,12:FOR I=1 TO 140:I
F PEEK(53279)=6 THEN RETURN
730 SOUND 1,254,10,10:NEXT I
740 FOR C=0 TO 3:SOUND C,0,0,0:NEXT C
750 ? #6:? #6;"       BOARD$=";PEEK(201
)
760 IF PEEK(53279)<>6 THEN 760
770 RETURN
780 POKE 77,0:FOR I=1 TO 4
790 READ X1,Y1,X2,Y2
800 IF X1=99 THEN RESTORE 1000:GOTO 78
0
810 PLOT X1,Y1:DRAWTO X2,Y2
820 NEXT I:SETCOLOR 1,INT(RND(0)*16),1
0
830 B=B+1:? :? "                BOARD #"
;B
840 RETURN
850 DIM PMMOV$(100):MOVE=ADR(PMMOV$):F
OR I=1 TO 100:READ N:PMMOV$(I)=CHR$(N)
:NEXT I
860 PMBASE=INT((PEEK(145)+3)/4)*4:POKE
 54279,PMBASE
870 PMB=PMBASE*256:POKE 559,46:POKE 53
277,3
880 DATA 104,162,0,232,142,10,212,142,
23,208,142,22,208,224,255,240,3,76,131
,6,200,192,255,240,3,76,131,6,96
890 REM *****
900 DATA 216,104,104,104,133,213,104,2
4,105,2,133,206,104,133,205,104,133,20
4,104,133,203,104,104,133,208
910 DATA 104,104,133,209,104,104,24,10
1,209,133,207,166,213,240,16,165,205,2
4,105,128,133,205,165,206,105
```

```
920 DATA 0,133,206,202,208,240,160,0,1
62,0,196,209,144,19,196,207,176,15,132
,212,138,168,177,203,164
930 DATA 212,145,205,232,169,0,240,4,1
69,0,145,205,200,192,128,208,224,166,2
13,165,208,157,0,208,96
940 REM ***
950 DATA 24,126,255,126,60
960 DATA 24,60,126,102,126,60
970 DATA 84,121,254,254,121,84
980 REM ***
990 DATA 28,4,16,4,16,4,5,15,5,15,35,1
5,38,1,24,15
1000 DATA 9,4,30,4,9,15,30,15,0,0,0,0,
0,0,0,0
1010 DATA 10,8,10,19,20,0,20,11,30,8,3
0,19,0,0,0,0
1020 DATA 20,18,20,13,1,10,17,10,20,1,
20,7,23,10,38,10
1030 DATA 7,7,7,12,7,12,32,12,32,12,32
,7,32,7,7,7
1040 DATA 1,6,20,6,30,6,38,6,20,12,38,
12,0,0,0,0
1050 DATA 1,10,10,1,38,10,30,18,10,10,
10,18,30,1,30,10
1060 DATA 8,9,16,4,9,10,16,16,32,10,26
,16,28,4,33,9
1070 DATA 20,4,20,15,6,10,33,10,9,15,3
0,15,9,4,30,4
1080 DATA 15,15,32,15,10,11,24,11,14,7
,28,7,19,3,32,3
1090 DATA 8,4,8,15,30,4,30,15,4,10,35,
10,13,15,25,4
1100 DATA 5,4,14,13,34,4,25,13,5,4,34,
4,19,1,19,10
1110 DATA 9,1,9,13,25,13,9,13,15,4,30,
4,30,4,30,18
1120 DATA 19,1,7,14,7,14,19,14,24,14,3
9,14,32,6,24,14
1130 DATA 11,1,13,3,17,7,24,14,5,14,38
,14,24,14,24,18
1140 DATA 1,4,29,4,5,9,38,9,11,15,23,1
5,24,15,34,5
1150 DATA 17,4,7,15,19,4,19,15,21,4,31
,15,0,0,0,0
1160 DATA 28,4,16,4,16,4,5,15,5,15,35,
15,38,1,24,15
1170 DATA 99,0,0,0
1180 RETURN
```

•

CHECKSUM DATA

(See page 23)

```
10 DATA 878,525,753,529,773,317,749,48
8,476,975,631,287,988,522,304,9195
160 DATA 84,919,642,170,60,749,319,504
,976,989,298,334,999,12,374,7429
310 DATA 318,24,312,19,44,2,15,314,342
,499,510,501,522,329,719,4470
460 DATA 786,240,543,795,827,37,101,58
8,80,751,607,543,490,763,591,7742
610 DATA 732,152,945,71,901,940,900,35
6,924,832,801,202,797,908,370,9831
760 DATA 862,614,341,663,169,924,81,24
6,607,463,527,709,13,803,636,7658
910 DATA 729,432,185,354,942,227,462,3
66,263,728,84,595,368,10,537,6282
1060 DATA 455,324,452,329,134,184,453,
514,154,297,370,161,790,4617
```

•

Omnitrend's Universe
by Thomas R. Carbone
and William G.M. Leslie III
Omnitrend Software
P.O. Box 3
West Simsbury, CT 06092
(203) 658-6917
$89.95

by Ray Berube

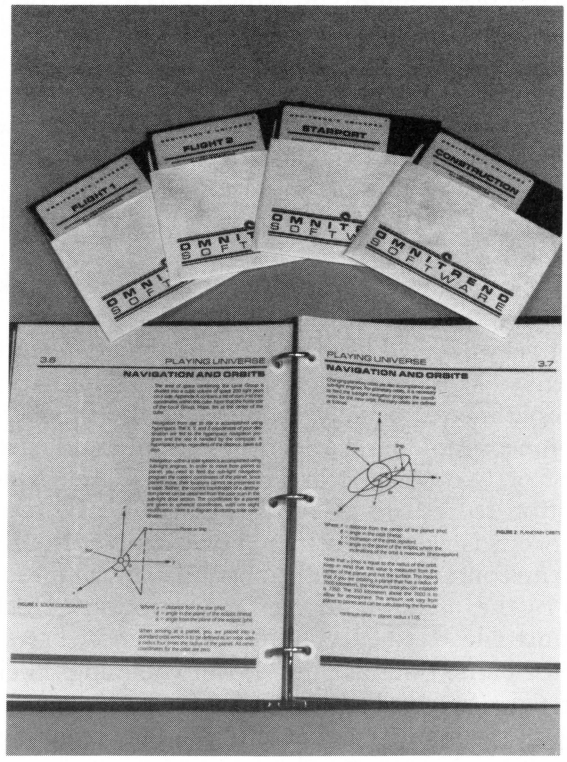
I first saw Omnitrend's **Universe** color advertisement in the pages of **ANALOG**. The test of the ad promised that "using high-resolution graphics, and more than 30 custom displays — distributed on four disks — Omnitrend's **Universe** allows you to experience the life of a starship captain. . ." As an avid role-playing gamer and **Traveler** fan, I was immediately intrigued. So, when asked if I could find the time to review **Universe**, I replied that I would *make* the time! Little did I know how much time would eventually be invested in reviewing this game.

Before plunging into the log of my five-year journey through the **Universe**, I'd like to call attention to some "cosmetic" features of the game, and then briefly outline the idea behind Omnitrend's **Universe**.

The first thing that sells a game is usually the price or the reputation of its designers. Here, Omnitrend takes a big gamble. The designers (and the first page of the manual lists Mr. Carbone, Mr. Leslie, and a host of others) are not familiar to me. Then again, neither is Omnitrend Software. Names like Infocom, Scott Adams, Adventure International Sierra On-Line, Activision, Carol Shaw, Larry Kaplan, Epyx and others ring a bell — and sometimes a gong — with adventure enthusiasts. So adding a hefty price tag and an unknown group of designers to a new product is taking a gamble. Omnitrend felt the game was worth the gamble, and they were right. **Universe** should help to establish Omnitrend and its designers. It will admit them to that privileged club of quality game producers.

One of the most important cosmetic aspects of a game is its packaging. Infocom recognized that early on, and Omnitrend has followed their example. **Universe** is beautifully presented in a thickly padded, self-standing binder. This binder contains the game's instructions, a manual of operations, a disk sleeve for each disk and ample room to include any documentation the player decides to add (and add it you will!). The folder is clearly divided and organized to facilitate referencing and play. The paper is a high quality, glossy magazine stock, and typesetting is clear and easy to read. I used looseleaf reinforcements on the pages to prevent tearing, and with all the use the pages are put to during play, I recommend it. Fine, you say, but nice packaging doesn't make a game. Agreed, but in this case it enhances the game's playability, so it is more than just a pretty box.

A word about the cost of **Universe** and then on to its playability. The game lists for $89.95, but some incidental expenses (which are suggested in the manual but considered necessary by this reviewer) will push the cost to over $100.00 for the game. I bought the reinforcements and a package of looseleaf filler, and suggest you do the same. In addition, the manual *suggests* that the player copy the disks to protect them from damage. I say it's necessary! You will need six blank disks: four to copy the game disks, a player disk you'll have to create and, finally, another disk to copy the player disk (more on why later). All this copying is made much less tedious by using Brian Moriarty's **Black Rabbit 2** (**ANALOG** Issue 9 or the improved version in the **ANALOG Compendium**). The progress of the game relies on menu selections, and disk swapping is constant. You'll cry if you don't copy a game disk and — while inserting it for the umpteenth time — you damage it. Your game is now worthless, so please make the copies!



Universe.

The premise of **Universe** is very simple. You are part of a fringe star group hundreds of light years from Earth. Your society depends on regular assistance packages from Earth which arrive via a one-way hyperspace booster system. Suddenly these packages stop arriving. Chaos and decline threaten your civilization, but hope springs out of confusion. It is

believed a hyperspace booster of similar design to those which allowed contact with Earth has surfaced in your star sector. Find the booster and you'll save your civilization.

Sounds easy, but wait! No one knows where the booster is, and as you begin the game, you don't even have a space ship. So you're a long way from grabbing that booster and saving civilization. This is where **Universe** begins and, right from its opening graphics, it grabs you.

The game allows you to enter into a mortgage and buy a ship and the barest minimum to outfit it. But star travel is expensive! In order to fund your search for the booster and keep your ship flying, you'll have to engage in mining or passenger transport or trading or contract work or even pirating innocent vessels. It is here in this realm of real experience that **Universe** overwhelms you. It is filled with detail and depth of experience. I can't imagine anyone playing **Universe** to the point of completely exploring all of its aspects! You can find the booster, but . . . that's just a lure to get you into becoming a starship captain. Once you own your ship, the game is exploration, gambling, warfare, experiencing a **Universe**.

How does Omnitrend achieve this wonderful feat? By a very structured and logical set of menus which guide the player but present new possibilities at each selection. What is a menu? Well, as a main frame business programmer, I'm very familiar with menu-driven data base systems, but it's not a common feature in adventures for the computer. So I'm going to break down the organization of **Universe** by delving into its menus.

Included in the appendices of **Universe** is a skeletal flowchart of how the game's logic proceeds. It is not very detailed but can be used to understand the menus. Most computer gamers are familiar with simple menus usually found after the title screen. Some examples are: select joystick or paddle; select one or two players; press start for a new game or option to restore a saved game. In **Universe** this concept is carried further. For example, the Flight Menu consists of nine options and many of these lead to further menus and even more options. Docking Control leads to Parts Removal, Purchase Parts and Place Parts. Sounds a little overwhelming? Well, at first glance it is. To play **Universe** a thorough reading of the manual and careful pre-play planning is a necessity. Now on to play action.

Play begins by booting the Construction disk, and after some impressive titles and opening music, the first block of text appears. This early text sequence doesn't allow for any real decisions from the player. You meet a bank loan officer, mortgage your ship, and are sent to the shipyards to choose a starship. From this point on, you are in control of your destiny! On arriving at the shipyards you encounter the first menu, a list of ten ship designs. Each design can be called up and examined in detail. The screen, through a graphics window, presents a three-dimensional view, an overview and a side view of the design while a text area provides statistics such as cost, size, visibility, integrity and specific features. It's a good idea to study each ship design carefully. Some are more suited to mining or pirating than others and a poor choice can spell disaster later in the game. After selecting a design, the game requires you to create a player disk. This is a tedious task comprised of disk swapping that lasts for more than ten minutes! As the manual suggests, be patient. There's a lot of data being transferred. As soon as you complete your player disk, copy it! Otherwise, should you meet with an untimely accident like death later on (in the game, I mean), you'll have to re-create your player disk. With a copy you can pick right back up with the next section: Flight One.

The Flight One disk in conjunction with the Flight Two disk contains all of the menus needed for space operations. You select locations for equipment installation, hire crew members, buy fuel, weapons, additional equipment such as scanners, rescue pods, etc. You must become familiar with the operation of every part of your ship. In order to enter hyperspace, you must understand how your drive works (and there are several drives to choose from). Here is where the menus allow you total access to every part of your ship. You can select to operate or study drives, weapons, scanners, mining systems, computer controls, orbital shuttle functions and more — from just ONE flight menu! The ship is yours to exploit fully.

From passenger transfers to orbital shuttle repairs, the Starport menu covers just about any activity that might take place on reaching a distant star system. I can't begin to delve fully into each area of these menus in this limited review. In fact, I've played **Universe** steadily, six to eight hours a day, for over a week to do this review, and I'm still discovering new elements of the game! The Starport menu allows you to buy, sell and trade goods and services. It contains the activities of customs clearance, transfer of passengers, even repair and fueling of your shuttle. Outside of some unsavory activities omitted, it covers all the ground needed for exciting starport intrigue.

To conclude, **Universe** is a blend of text and graphics adventure. Though in places its pace is slower than most shoot-em-ups or text adventures, the richness of detail is unmatched. And when you're in a fire fight, nothing happens slowly.

If you buy **Universe**, you'll find months of playing pleasure ahead of you. Even though its price tag is a bit hefty, you get your money's worth and more. Let's face it, if we can shell out forty bucks for **Dig/Dug** and be bored with it after a week, we can surely invest $89.95 or so for a game we'll still be playing throughout the year. □

Bacterion!

16K Cassette or 24K Disk

by Kyle Peacock with Tom Hudson

The year is 2284. The "Robotron Incident" of 2084 has long been forgotten, and the field of genetic engineering has led to the fabrication of synthetic humans. Each fabricated unit is genetically engineered to perform a specific task to perfection.

Fifty years later, through a unanimous vote of the United Nations, these "Syntrons" have been installed into every top political position in the world. Each represents the opinion of the particular government they were assigned. The fate of continents rests in the hands of the Syntrons.

Now the year is 2369. Long-term studies of the Syntrons' behavioral patterns yield strange results. A genetically invulnerable bacteria has infected the species. The bacteria preys on the cerebral nerves responsible for all higher order brain processes, making the individual deranged and quite unstable.

This bacteria has begun to trickle down to infect the Syntrons' counterpart: man. Soon, mankind will teeter on the very brink of insanity and demise, unless a means can be found to halt the plague of 2369: **Bacterion!**

Loading instructions.

Before typing anything, look at the listings accompanying this article.

Listing 1 is the BASIC data and data checking routine. This listing is used to create both cassette and disk versions of **Bacterion!** The data statements are listed in hexadecimal (base 16), so the program will fit in 16K cassette systems. This makes typing more difficult, but if you want to play the game...

Listing 2 is the assembly language source

Expand Your Atari ...With peripherals from MAP

~~$199.00~~
New Low Price
$149.95

MAP-1000C Modem

- Auto Answer / Auto Dial
- Direct Connect to Phone Line
- No Atari 850™ Interface Module Needed
- Includes AC Adapter / Power Supply
- Free CompuServe DemoPak™
- 1 year warranty
- Connects to Joystick Port
- Works on **ALL** Atari Computers

SOPHISTICATED SMART TERMINAL SOFTWARE ON CARTRIDGE

FEATURES:

- Supports XMODEM Protocol
- ASCII / ATASCII Translation
- Allows Transfer of Files Larger than Memory
- Upload / Download of Text and Programs
- 100% Machine Language
- Multiple Buffers
- Off-Line Editing
- Variable Baud Rate
- Parity Options
- Full / Half Duplex

MAP-1150 Printer Interface

Only $99.95

- Replaces Atari 850™ Interface Module
- Compatible with all software
- 3 foot cable with Centronics plug (compatible with Epson, NEC, Prowriter, etc.)
- 2 year warranty
- Connects to serial bus on computer
- Daisy chains with other Atari peripherals
- Works on **ALL** Atari Computers

MAP
MICROBITS PERIPHERAL PRODUCTS
225 W. Third St. / Albany, OR 97321 / (503) 967-9075

CIRCLE #121 ON READER SERVICE CARD.

code for **Bacterion!** You *do not* have to type this listing to play the game! It is included for those readers interested in assembly language.

Follow the instructions below to make either a cassette or disk version of **Bacterion!**

Cassette instructions.

1. Type Listing 1 into your computer using the BASIC cartridge, and verify your typing with **C:CHECK** (see page 23).

2. Type RUN and press RETURN. The program will begin and ask:

MAKE CASSETTE (0) OR DISK (1)?

Type 0 and press RETURN. The program will begin checking the DATA statements, printing the line number of each as it goes. It will alert you if it finds any problems. Fix any incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all DATA lines are correct, the computer will beep twice and prompt you to "READY CASSETTE AND PRESS RETURN." Insert a blank cassette in your recorder, press the RECORD and PLAY buttons simultaneously and hit RETURN. The message "WRITING FILE" will appear, and the program will create a machine language boot tape version of **Bacterion!**, printing each DATA line number as it goes. When the READY prompt appears, the game is recorded and ready to play. CSAVE the BASIC program onto a separate tape before continuing.

4. To play the game, rewind the tape created by the BASIC program to the beginning. Turn your computer OFF and remove all cartridges. Press the PLAY button on your recorder and turn ON your computer while holding down the START key. If you have a 600 or 800XL computer, you must hold the START and OPTION keys when you turn on the power. The computer will "beep" once. Hit the RETURN key and **Bacterion!** will load and run automatically.

Disk instructions.

1. Type Listing 1 into your computer, using the BASIC cartridge, and verify your typing with **D:CHECK2** (see page 23).

2. Type RUN and press RETURN. The program will ask:

MAKE CASSETTE (0) OR DISK (1)?

Type 1 and press RETURN. The program will begin checking the DATA lines, printing the line number of each statement as it goes. It will alert you if it finds any problems. Fix incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all DATA lines are correct, you will be prompted to "INSERT DISK WITH DOS, PRESS RETURN." Put a disk containing DOS 2.0S into drive #1 and press RETURN. The message "WRITING FILE" will appear, and the program will create an AUTORUN.SYS file on the disk, displaying each DATA line number as it goes. When the READY prompt appears, the game is ready to play. Be sure the BASIC program is SAVEd before continuing.

4. To play the game, insert the disk containing the AUTORUN.SYS file into drive #1. Turn your computer OFF, remove all cartridges and turn the computer back ON. **Bacterion!** will load and run automatically.

Playing the game.

Bacterion! is a game for one or two players. In the two-player mode, both play simultaneously. In this mode, the game is more cooperative than competitive. You must prevent the six (count 'em, six!) different strains of Bacterion from removing the ten cerebral cells from the host Syntron. A cell is inoperative when it is completely removed from the screen.

(continued on page 39)

ATARI®

ATARI is a trademark of ATARI, INC.

NEW LOWER PRICES!!

ATARI 800XL ..... Call

Atari 1050 Disk Drive ..... Call
Atari 1010 Recorder 77.00
Atari 1027 Printer ..... Call
Atari 850 Interface ..... Call

MOSAIC

64K Ram/400 ..... 149.00
64K Ram + Cable Kit/400/800 ..... 169.00
48K Ram Kit 94.00
16/32 Expander 64.95
32K Ram 77.95
Mosaic Adaptor 49.95

Memory Expansion for Atari 600XL .. Call

DISK DRIVES

Percom .....
Trak ..... Call
Rana ..... For
Indus Gt .. Prices

NEW

RS232 Modem Adaptor (Through serial port) 39.95

Koala Touch Tablet Cart or Disk ... 69.95

MONITORS

We carry a full line of AMDEK and USI Monitors.

LET YOUR ATARI IMPROVE YOUR LIFE AND YOUR MIND

NEW - THE STIMUTECH SUBLIMINAL PROGRAM

With the **Expando-Vision Interface** the Atari Computer user can now work on improving his or her personal environment during normal television viewing with a proven psychological technique that was banned by the F.C.C.

Buy the Expando-Vision Interface for $99 and get the Rom Cart of your choice for **Free!!!**

Rom Carts are available for:

Weight Control
Study Habits
Stress Control
Drinking Control
Smoking Control
Career Success
Sexual Confidence

Additional Rom Carts $29.95 each

3RD Party Printer Interfaces

Interfast I 139.00
Apeface w/cable included 69.95

PRINTERS

SUPER SPECIAL - Prowriter Package with Apeface or Interfast I ..... Call
Axiom AT-100 ..... 229.00
Axiom AT-550 ..... 329.00

ATARI

Programming 2 & 3 - T 23.95
Conversational Lang. - T ... 44.95
Music Composer - Cart ... 32.95
My First Alphabet - D 26.95
Touch Typing - T 19.95
Home Filing Mgr. - D 37.95
Mailing List - T 19.95
Caverns of Mars - D 28.95
Computer Chess - Cart ... 26.95
Missile Command - Cart ... 26.95
Super Breakout - Cart 26.95
Star Raiders - Cart 32.95
Assembly Editor - Cart 46.95
Basic - Cart 41.95
Macroassembler - D 67.95
Pilot (Home Pkg.) - Cart ... 58.95
Invitation to Program I T ... 19.95
Speed Reading - T 55.95
Basketball - Cart 26.95
Graph-it - T 15.95
Juggles House - D/T 22.95
Pilot (Educator) - Cart 97.95
Video Easel - Cart 26.95
Defender - Cart 32.95
Galaxian - Cart 32.95
Qix - Cart 32.95
Dig Dug - Cart 32.95
ET - Cart 37.95
Timewise - D 23.95
Atariwriter - Cart 74.95
Donkey Kong - Cart 37.95
Ms. Pac-Man - Cart 39.95
Tennis - Cart 35.95
Eastern Front - Cart 32.95
Donkey Kong Jr. - Cart ... 39.95
Pengo - Cart 35.95
Logo - Cart 79.95
Robitron - Cart 35.95
Pole Position - Cart 39.95
Microsoft Basic II - C 67.95
Paint - D 33.95
Caverns of Mars - Cart 32.95
Joust - Cart 39.95
Visicalc - D 159.95

THORN

War Games - Cart 27.95
Hockey - Cart 27.95
Soccer - Cart 30.95
Submarine Commander - Cart ... 30.95

SIERRA ON-LINE

Homeword - D ..... Call
Dark Crystal - D 27.95
Frogger - D/T 23.95
Quest For Tires - D 23.95
Ultima I - D 23.95
Ultima II - D 41.94
Wizard/Princess - D 22.95

SPINNAKER

Up For Grabs - Cart 27.95
Cosmic Life - Cart 23.95
Facemaker - Cart/Disk 23.95
Alphabet Zoo - Cart 23.95
Fraction Fever - Cart 23.95
Kids on Keys - Cart 23.95
Delta Drawing - Cart 27.95
Story Machine - Disk 23.95
Rhymes & Riddles - D 20.95
Amazing Things - D 27.95
Hey Diddle Diddle - D 20.95
Snooper Troops 1 or 2 - D .. 30.95
Trains - D 27.95
Aerobics - D 30.95

SIRIUS

Bandits - D 23.95
Way Out - D 27.95
Repton - D 27.95
Twerps - D 23.95
Blade of Blackpoole - D .. 27.95
Type Attack - D 27.95
Beer Run - D 23.95
Critical Mass - D 27.95
Wavy Navy - D 23.95

GAMESTAR

Baja Buggies -D/T 22.95
Starleague Baseball - D/T 22.95
Starbowl Football D/T ... 22.95

SEGA

Buck Rogers - Cart 27.95
Star Trek- Cart 27.95
Congo Bongo- Cart 27.95
Tac Scan- Cart 27.95

EPYX

Gateway To Apshai - Cart 27.95
Pitstop - Cart 27.95
Lunar Outpost - Cart 27.95
Swat Rescue - Cart 27.95
Silicon Warrior - Cart 27.95
Fun With Music - Cart 27.95
Jumpman Jr. - Cart 27.95
Dragonriders of Pern - D/C 27.95
Temple of Apshai - D/C ... 27.95
Hellfire Warrior - D/C 20.95
Fun With Art - Cart 27.95

INFOCOM

Deadline - D 34.95
Witness - D 34.95
Zork I, II, III - D 27.95
Starcross - D 27.95
Planetfall - D 34.95
Suspended - D 34.95
Enchanter - D 34.95
Infidel - D 34.95

Hundreds of items available for the ATARI ... please call.

LJK

Letter Perfect - D 74.95
Data Perfect - D 74.95
Spell Perfect - D 64.95

BRODERBUND

AE - D 23.95
Arcade Machine - D 41.95
Bank St. Writer - D 49.95
Lode Runner - D 23.95
Drol - D 23.95
Spare Change - D 23.95
Choplifter - D 23.95

DATA SOFT

Money Wizard - D 49.95
Letter Wizard - D 49.95
Spell Wizard - D 34.95
Genesis - D/T 20.95
Heathcliff - D/T 20.95
Pooyan - D/T 20.95
Zaxxon - D/T 27.95
Micropainter - D 23.95

SYNAPSE

Blue Max - D/T 23.95
Dimension X - D/T 23.95
Drelbs - D/T 23.95
Encounter - D/T 23.95
Necromancer - D/T 23.95
Quasimoto - D/T 23.95
Rainbow Walker - D/T ... 23.95
Shamus Case II - D/T ... 23.95
Zeppelin - D/T 23.95
N.Y.C. - D/T 23.95

FIRST STAR

Bristles - D/T 20.95
Flip Flop - D/T 20.95
Astro Chase - D 20.95
Boulder Dash - D/T 20.95

STRATEGIC SIMULATIONS

Carrier Force - D 41.95
Combat Leader - D/T ... 27.95
Battle For Normandy -D/T 27.95
Eagles - D/T 27.95

PARKER BROS.

Astrochase - Cart 34.95
Frogger - Cart 34.95
Q*Bert - Cart 34.95
Popeye - Cart 34.95

MISCELLANEOUS

Castle-Wolfenstein - D .. 20.95
Home Accountant - D ... 52.95
Master Type - D/Cart ... 27.95
Spelunker - D 27.95
Flight Simulator II - D ... 37.95
Zombies - D/T 23.95
Mr. Robot - D 23.95
River Raid - Cart 31.95
Diskey - D 34.95
Chatterbee - D 27.95
Rally Speedway - Cart ... 34.95
Ultima III - D 41.95
Miner 2049'er - Cart 34.95
Scraper Caper - Cart ... 34.95
Basic XL - Cart 74.95
Monkey Wrench II - Cart . 49.95
Omnimon 82.95
Scraper Caper - Cart ... 34.95

BOOKS

Compute's First Bk/Games 12.95
Inside Atari DOS 19.95
Mapping the Atari 14.95
Machine Language/ Beginners 14.95
Your Atari Computer ... 16.95
The Atari Assembler 12.95
Visicalc Book-Atari 14.95

ACCESSORIES

WICO Joystick ..... Call
Flip 'n' File Trunks 20.95
Flip 'n' File Cart 20.95
Joysensor 24.95
Elephant Disks (Box of 10) 20.00
WICO Trakball 37.95
KRAFT Joystick 15.95

D - Disk T - Cassette
Cart - Cartridge

ComputAbility™

To Order Call Toll Free
800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders -
414-351-2007

NO SURCHARGE FOR MASTERCARD OR VISA

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

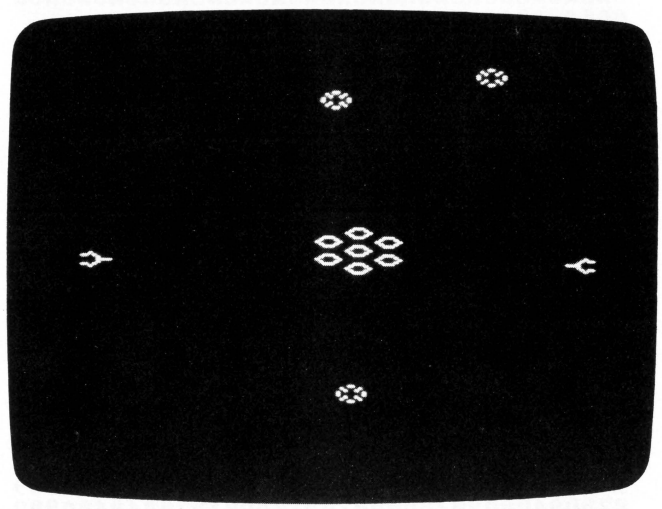
MasterCard VISA

ORDER LINES OPEN
Mon-Fri 11 AM - 7 PM CST
Sat 12 PM - 5 PM CST

ORDERING INFORMATION For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 3% shipping on all Hardware orders, minimum $3.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders - add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

CIRCLE #123 ON READER SERVICE CARD.

As a member of BARF (Biological Armament Restriction Force), you are equipped with a miniaturized GEV (Genetic Extermination Vehicle) to aid you in your mission. The vehicle emits high-frequency photons from its nose, capable of vaporizing the Bacterion, while leaving brain tissue and other GEV's undamaged.



Bacterion!

Control over your GEV is accomplished through the joystick. A two-player game will require as many joysticks. Pushing up will cause your GEV to move forward. Moving the stick left or right will cause the vehicle to rotate in the respective direction. Pushing the trigger fires the photonic bursts. Any one player can have up to four high-energy photons on the screen at once. Holding down the trigger fires all available photons in rapid succession. Should your GEV be destroyed by either running into your opponent or being skragged by a Bacterion, you must wait several seconds for another GEV to be injected into the host. Parking or driving your GEV over a brain cell gives it unsure traction, which may cause the vehicle to rotate randomly. Each GEV is internally powered, so there is an unlimited supply of ammunition.

Initially, each Syntron has a unique brain pattern. You may select which of the three top political figures you wish to put "under the knife." Each individual has a total of ten cerebral cells. Some cells may be harder to visualize, since there may be two or more on top of one another.

The SELECT button will choose between a one- or two-player game. The OPTION button will choose which political leader you will operate on. Examination of the speed at which the colors change in the word "BACTERION!" on the title screen dictate the brain pattern. The Prime Minister of England, President of the U.S.S.R. and the President of the U.S. are your three available candidates. The START button will begin the game. Pausing or examination of scores is accomplished by pressing the space bar. Pressing the space bar a second time will resume the life-or-death struggle. Each of the six attacking strains are worth from 10 to 60 points, respectively.

Additional credits.

My special thanks goes to Tom Hudson for his assistance in **Bacterion!** Through the use of his Graphics 7+ handler presented in issue #11, and some custom-made shape-drawing routines, we've put together the best (as in #1) two-player public domain assembly language game *ever*! If you agree or disagree, I'd like to hear it. Drop me a line, care of Kyle Peacock here at the editorial offices of **ANALOG.**

BASIC Listing.

```
10 REM *** BACTERION ***
20 TRAP 20:? "MAKE CASSETTE (0), OR DI
SK (1)";:INPUT DSK:IF DSK>1 THEN 20
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,0:PUT #1,40:CLOSE #1:END
160 FOR X=1 TO 2:PUT #1,0:NEXT X:CLOSE
 #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,40:PUT #1,85:PUT #1,59:GOTO 210
200 ? "READY CASSETTE AND PRESS RETURN
";:OPEN #1,8,128,"C:":RESTORE 230:FOR
X=1 TO 40:READ N:PUT #1,N:NEXT X
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
230 DATA 0,39,216,39,255,39,169,0,141,
47,2,169,60,141,2,211,169,0,141,231,2,
133,14,169,56,141,232,2
240 DATA 133,15,169,0,133,10,169,40,13
3,11,24,96
1000 DATA 2065E4A9228D2F02A9288581A900
8580A9088583A9008582A000B1809182C8D0F9
E681E683A581C940D0EFA90B,831
1010 DATA 850D8503850BA912850C8502850A
4C120BA200A90E9D001FCAD0FAA9708D001F8D
011FA9F08D021FA94E8D031F,796
1020 DATA 8D6B1FA9208D051FA9108D041FA9
308D6D1FA9008D6C1FA9418DC81FA98E8DC71F
A91F8DCA1FA9008DC91FA93E,396
1030 DATA 8D2F02A9008D07D4A9038D1DD0A9
108D6F02A900AA9D00049D0005CAD0F7A202A9
0095A995AC95AFCA10F7A910,369
1040 DATA 8580A9208581A200A5809D351BA5
819DF51BA900A02791808810FBE8E0C0F010A5
80186928858 0A58169008581,840
```

```
1050 DATA 4CBC08A6A4BC7D09A20986B7B980
0995E895D4B99E099DC91C9DB51CA90095C0C8
CA10E8A20086A88E7D1D8E7E,608
1060 DATA 1D204209A5A5F004E8204209A901
8D451E8D1F1E8D201E8D211EA205BD71099DAB
13BD6B099DA413BD77099DD9,837
1070 DATA 0ACA10EB4CBC09A9009D691D9D1D
1EA9019D461E9D5A1EBD69099D311E9D2D1DBD
67099D7D1DA9809D911D602C,103
1080 DATA C30C04342C241C140C0D0B090705
0346413C37322D000A1444505C3E4A56624450
5C4F4F4F4F4955435B3D614F,598
1090 DATA 47574F4F4F4F47574F5D5D5D6060
60606363634C52585E626266666A6A565A5A5E
5E5E5E626266A201869ECA8A,956
1100 DATA 8D05D28D07D28D08D29D00039D00
069D00078D1ED0CAD0F1A204AD451E9D1D1E20
EB0A9D7D1DAD0AD21004A915,391
1110 DATA D002A9E49D911D20F40AA9008D0F
0B85B69D691D9DDD1C9DCD1D9DF11C9D551DAD
FD0AC9D8B057A90A20FD0AA8,809
1120 DATA 8E100BBD551DF00698DDDD1CF03B
E8E005D0F0AE100BB9B51CC9C2B02CA90220FD
0A8C110BA8B9E10A9D551DB9,245
1130 DATA DF0AAC110B1879B51C9DB91DB9D4
0018692C9DA51D989DDD1C4C810AAE100BCE0F
0BD0A9A4A5C89820FD0AA8B9,236
1140 DATA 7D1D9DA51DB9911D9DB91DCAE001
F0034CDC09A92885B5AE451EBDD80A85B9C905
900638E9039DD80AA20CA9FF,604
1150 DATA 9D6E1E95D99DBA1C95C5CAE004D0
F1A203AD0AD229F0090A9DC002AD0AD229F01D
E70A9DC402CA10E8A90085BC,311
1160 DATA 85BE85BD85BF859E600000000000
0015240AF50C0C0A0A0C08080AA96420FD0A18
694B60A91020FD0A9D2D1D60,732
1170 DATA 8D0E0BAD0AD2CD0E0B90054A4A4C
030B6000000000A9008D01D28D03D220E30E20
430FA90185A8859EA906A20C,377
1180 DATA A0F8205CE4A907A20DA056205CE4
A9C08D0ED420BA0FA5A3C901D0F7E6A3203E08
A5A3C902D0C1ADFC02C921D0,328
1190 DATA 37AD0FD22904F03020430FA5B748
A90A85B7A90185A8859E8DFC0220BA0FADFC02
C921D0F9AD0FD22904F0F968,687
1200 DATA 85B7A90085A8859E8DFC02C6B310
04A90985B3A6B3B5C0D012B5E88588BDC91C85
89A901858AA90020401AA909,270
1210 DATA 85B2A6B2B5C0F00320101AC6B210
F3A9048DB70DAEB70DBDCD1DF053302E09809D
CD1DA902858ABD7D1D38E92C,616
1220 DATA 9DF51D8588BD911D38E91CC9C0B0
2E9D091E8589BDCD1D297F20401A4C200CDEE1
1D101EBDF51D8588BD091E85,525
1230 DATA 89A900858ABDCD1D297F20401AAE
B70DA9009DCD1DCEB70DADB70DC901D09BA211
B5C0F033B5E88588BDC91C85,157
1240 DATA 89A90095C0858A8E6E10204219AE
6E10B5D495E88588BDB51C9DC91C8589A90385
8A8E6E10204219AE6E10CAE0,423
1250 DATA 09D0C4A5B7101320BA0FA901859E
8514A613E8E413D0FC4C120BA5B6C903F0034C
4B0B20430FEE451EAD451EC9,175
1260 DATA 079051AD0AD229F08DC802A90185
9E85A820BA0FA5B748A90A85B78514A613E8E4
13D0FC6885B7A90085A8859E,792
1270 DATA 8DFC0285A085A1859FA205BDAB13
C9029003DEAB13BDA413C908900638E9049DA4
13CA10E6A9018D451E8D1F1E,798
1280 DATA 8D201E8D211E20BC094C4B0BD8A5
A80AAABD72108D3002BD73108D3102BD76108D
0002BD77108D0102A9008D00,591
1290 DATA D08D01D08D02D08D03D08D04D08D
05D08D06D08D07D0A59ED020854DADC4021869
108DC402A90485B4202011C6,663
1300 DATA B4A5B4C901D0F52020122076124C
5FE4D820BB0E207F0DA59ED01B201417205E18
208510209C1720961220CE0D,739
1310 DATA 20FE0D20220E20B2134C62E4A6B7
3019BDB80D85A2BDC30D85A7A6A28E00D2E88E
02D2A5A6F003C6A660A5A785,314
1320 DATA A6EEB60DADB60D2901AABDB40D8D
01D28D03D26000A40000A0AAB4BEC8D2DCE6F0
FA8C020406080A0C0E101214,141
1330 DATA 01A69FF00FCAC69FBDE20D8D04D2
BDF00D8D05D26000ECD8C5B19D8A76634F3B28
140100A2A2A2A2A4A4A4A4A4,963
1340 DATA A4A6A6A6A6A1F00FCAC6A1BD120E
8D06D2BD1A0E8D07D26000FED4AA7F552B0100
A8A8A8A8A8A8A8A8A6A0F00FCA,880
1350 DATA C6A0BD360E8D04D2BD790E8D05D2
6000FDF9F5F1EDEAE6E2DEDAD6D3CFCBC7C3BF
BCB8B4B0ACA8A5A19D999591,399
1360 DATA 8E8A86827E7B77736F6B6764605C
5854504D4945413D3936322E2A26221F1B1713
0F0B0804008A8A8A8A8A8A8A,57
1370 DATA 8A8A8A8A8A8A8A8A8A8A8A8A8A8A
8A8A8A8A8A8A8A8A8A8A8A8A8A8A8A8A8A8A8A
8A8A8A8A8A8A8A8A8A8A8A8A,267
1380 DATA 8A8A8A8A8A8A8A8A8A8A8A8A8AAD
1FD0C907F0048D180F60AD180FC906F009C905
F00AC903F03360A2014C120F,188
1390 DATA E6A5A5A5290185A5A213A9009D88
169D9C16CA10F7A6A5BD190FAABD1B0F9D8816
BD2F0F9D9C16CA10F14C120F,148
1400 DATA E6A4A5A4C903B0F8A20086A38E18
0F60000A13B0ACA1B9A5B200D100000000B0AC
A1B9A5B200D2005050505050,157
1410 DATA 5000000000000050505050505000
F8A6A5B4BEF01BAD451E0A0A0A0A1875AD95AD
A90075AB95ABA90075A995A9,431
1420 DATA 88D0E5CA10DEA205A4A58A398310
AAB5A948290F0950BC7A10999D16684A4A4A4A
0950999C16CA10DF18A204A0,288
1430 DATA 02B5A975AA99AF00CACA8810F4A2
02BC8010B5AF48290F0950997516684A4A4A4A
0950997416CA10E6D860A900,125
1440 DATA 8D05D28D07D2859F85A085A160D8
48AD7D1D8D00D0AD7E1D8D01D0AD7F1D8D02D0
AD801D8D03D0AD811D8D07D0,505
1450 DATA 1869028D06D01869028D05D01869
028D04D0A9078D0002A9108D0102684048A900
8D00D08D01D08D02D08D03D0,419
1460 DATA 8D04D08D05D08D06D08D07D06840
488A48A207AD7010BD6F10AD6F108D0AD48D17
D0AD6F10186902BD6F10CA10,977
1470 DATA EBA9658D0002A9108D0102CE7110
100EA5A48D7110EE71100E7110EE701068AA68
4048ADC5028D17D068400000,220
1480 DATA 0000001FEB10C90F2410010D030F
05110709080E0FA204E002B01CBD911DCDC810
B00FBC2D1DB9CB109D2D1D20,380
1490 DATA F7124CA710CDCA10B0ECBD7D1DCD
C710B00FBC2D1DB9DB109D2D1D20F7124CC310
CDC910B0ECCA10C1602C20C8,807
1500 DATA D708070605040302010000F0E0D0C
0B0A09000F0E0D0C0B0A0908070605040302010
70707070468816469C167070,410
1510 DATA 70F0463816B0464C165046B01620
46C4162046D8162046EC167070460017707046
6016467416 41EB10A6B4BD69,515
1520 DATA 1DF00160BDDD1C3039A4A5B9691D
D02FBD7D1D38F97D1D20E112C5B5B021BD911D
38F9911D20E112C5B5B013B9,866
1530 DATA 7D1D8D7F13B9911D8D8013204013
A901D0118810C9BDA51D8D7F13BDB91D8D8013
A9009D411DAD7F13DD7D1DF0,532
1540 DATA 12B022AD8013DD911DD004A904D0
74B04A9035AD8013DD911DF066B004A900F062
A908D05EAD8013DD911DD004,781
1550 DATA A90CD052B03B9000200512F00690
08A90DD043A90ED03FA90FD03B200512F00690
08A903D030A902D02CA901D0,128
1560 DATA 28200512F0069008A905D01DA906
D019A907D015200512F0069008A90BD00AA90A
D006A909D002A9FF9D191D60,208
1570 DATA AD801338FD911D20E1128D8313AD
7F1338FD7D1D20E112CD831360C6B81047AE45
1EBDA31385B8A204BD691DD0,212
1580 DATA 33A90320FD0AA8C8BD191DDD2D1D
F024B00E206C12C908B0109849FFA8C8300920
6C12C908F002B0F098187D2D,937
1590 DATA 1D20E9129D2D1DCAE001D0C360BD
191D38FD2D1D4CE112A204DE051D1013AC451E
B9AA13BCDD1C10020A0A9D05,90
1600 DATA 1D20F712CAE001D0E360A204BD69
1DD03EBDDD1C1016BD911DC90F9004C9EB900B
E6B6C6B7A9019D691DD023BD,613
1610 DATA 411DD01EBD191D1019BD551DF014
9DB91D20EB0A9DA51D20F40ABDDD1C09809DDD
1CCAE001D0B860100549FF18,346
1620 DATA 6901601004186910 60C910900338
E91060BD691DD043BD191D303EBC2D1DB98413
8D8113187D7D1D9D7D1DB994,359
```

```
1630 DATA 138D8213187D911D9D911DBDDD1C
101C297FA8AD81131879D40099D400AD821318
79B51C99B51CA90199C00060,832
1640 DATA DEF11C1029A5B99DF11CBDCD1DD0
1FBD191D4820771 1A8689D191DC0FFF00FB96F
139DCD1DA90A9DE11DA90E85,98
1650 DATA 9F60010808080406060602070707
0305050500000000000FFFEFEFEFEFEFF0001
020202020201FEFEFEFF0001,251
1660 DATA 0202020202010OFFFEFE00000000
00000000000000000000A9048D5814AE5814BC
1D1EBD311ED966149006B965,19
1670 DATA 149D311ED9651490F50AA8B97814
8584B97914858 5AD58140AA8B96E148586B96F
148587A9078D5A14187D911D,737
1680 DATA 8D5B14AC5A14B184AC5B149186CE
5B14CE5A1410EEA9078D5914AE5814BD911DAE
5914187D5D14A8A9009186CE,467
1690 DATA 591410E9CE5814108DCE5C141028
A9048D5C14AAE002B007BD1D1EC907D014BD31
1EC92FF00DFE311EC92ED006,922
1700 DATA E0029002E6B6CA10DE6000000000
03FCFDFEFF08090A0B0010141 81C1F23273000
040005000600070003D814E0,23
1710 DATA 14E814F014F81400150815101518
1520152815301538154015481550155815601 5
6815601570157815801578 15,526
1720 DATA 8815901598159015A015A815B015
B815C015C815C015D015D815E015E815F015F8
150016081610161816201628,329
1730 DATA 1630161010103854824444202010
1E192120108040201F1910101800 00C6391810
100C040B10F0100B04000C10,80
1740 DATA 101839C60000181010191F204080
102021191E10202044448244381010100804 84
9878080404180808 98F80402,243
1750 DATA 0130080818 9C630000 20D0080F08
D020000000631C1808083 0010204F898080818
0404087898840408 18002481,381
1760 DATA 8124001818422481812442189942
2481812442990000 3C24243C00000066421818
426600C3810018180081C33C,180
1770 DATA 42A58181A5423C00182442422418
000000183C3C1800000000 1824241800000018
005A5A0018001800 18A5A518,130
1780 DATA 00182020E4181827040400246618
1866240004042718 18E420200C409025258840
0C180099A1041842243 00219,301
1790 DATA 84A0190230244208 24A591001800
0000181800000000000 8381C10000000080878
1E10100008082CE007341010,877
1800 DATA 084A24C0032452108942248001 24
4291814200000000428181000000000008100
000000000000000000000000,587
1810 DATA 626163746572696F6E4100000000
0000B4A8A500B0ACA1A7B5A500AFA600D2D3D6
D900000000000 0B4A5A1AD00,10
1820 DATA B3A3AFB2A500000000000000000000
000000000000000000000000000000000000000
000000000000000000000000,870
1830 DATA 000000000000000000000000000000
000000000000000000000000000000000000000
002239000000000000000000,961
1840 DATA 00000000EBF9ECE500F0E5E1E3EF
E3EB000000000000000000000003729342800
000000000000000000000000,720
1850 DATA F4EFED00E8F5E4F3EFEE00000000
000000A1AEA1ACAFA700A3AFADB0B5B4A9AEA7
0000A6A5BD691DF00BDE691D,694
1860 DATA D0262042094C4617204A17DE7F17
1018A9039D7F17BC7802BD2D1D18798C1720E9
129D2D1D9D311ECA10CD60DE,608
1870 DATA 461E102FBD7802C90EF008C90AF0
04C906D006DE5A1E4C6717FE5A1EBC5A1EB981
179D5A1E9D461EBD461ECD8B,151
1880 DATA 17F00320F7126000000202020304
0506070808080000000000FFFFFF0001010100
000000A6A5BD691DD077A901,784
1890 DATA 8D5B18BD0CD02902F003202118EE
5B18BD0CD02904F003202118EE5B18BD0CD029
08F003202118EE5B18A00 38A,760
1900 DATA 1869013908D0F0032021188810F1
CE5D181028A9038D5D18BD04D02901F01CA902
20FD0AAC0AD2100549FF1869,744
1910 DATA 01187D2D1D20E9129D2D1D9D311E
BD04D02902F0068E5B18202118CA10818D1ED0
608C5C18AC5B18B9691DD00D,659
1920 DATA 8AA8203D18AC5B18203D18F6BEAC
5C1860B9691DD013A907991D1EA97899691DA9
00859FA94285A0606868AC5C,373
1930 DATA 186000000002064184CCA18A6A5BD
691DD05BB5BAF005D6BA4CC618B5BCC904B04C
BD10D0D047F6BCA90785A1A9,805
1940 DATA 0395BAA00CB96E1E300388D0F88C
40198A996E1EBC2D1D9848B984130A187D7D1D
38E92CAC4019997D1D68992D,281
1950 DATA 1DA8B994130A187D911D38E91CAC
401999911DCA109D60A20CBD6E1E306920F712
BD7D1DC9A2B03E95D9BD911D,165
1960 DATA C9C0B0359DBA1CA004B9691DD043
BD7D1D18693038F97D1DC909B035BD911D1869
2038F9911DC909B027203D18,879
1970 DATA 8E4119BD6E1EAAF6BEAE41198E41
19BD6E1EAAD6BCAE4119A9FF9D6E1E95D99DBA
1CD00588C001D0B3A90195C5,610
1980 DATA CAE004D08D600000A689E0C1B02E
BD351B8580BDF51B8581A588C9A0B01E2903AA
A5884A4A8598A48ABD2D1B39,116
1990 DATA 2518B595BD291BA4983180059591
8060A58CC589900B38E5898590A9018592D00B
A58938E58C8590A9FF8592A5,768
2000 DATA 8BC588900B38E588858FA9018591
D00BA58838E58B858FA9FF8591A900858E858D
A58FC590900A859385944A85,751
2010 DATA 8E4CCE19A5908593859 44A858DA5
93F03DA58E186590B009858EC594900EA58E38
E594858EA589186592858 9A5,593
2020 DATA 8D18658FB009858DC594900EA58D
38E594858DA5881865918588204219C693D0C3
60A6B2A90095C0858AB5E885,455
2030 DATA 88BDC91C8589A9002040 1AA6B2B5
D4858895E8BDB51C85899DC91CA901858AA900
20401A600A0A0A8599A90885,465
2040 DATA 9AA699BCDD1A3034B9851A859BB9
8D1A859CBD951A859DA58818659B8588A58918
659C8589A59AC908F0032042,972
2050 DATA 19A59DF004C69DD0E1E699C69AD0
C560000001FF01FF01FFFF010000FF0101FF03
0202020202020000280000000,178
2060 DATA 0000000028000000000000002800
0000000000002800000000000002800000000
0000002800000000000028,418
2070 DATA 0000000000000028000000000000
0205030704020 6FF0000FF00000000000001FF
00000000000002FF00000000,510
2080 DATA 000003FF0000000000004FF0000
0000000005FF000000000000 6FF0000000000
0007FF00000000000055AAFF,320
2090 DATA 3FCFF3FCC0300C03401004010000
000000000000000000000000000000000000000
000000000000000000000000,425
```

•

CHECKSUM DATA

(See page 23)

```
10 DATA 58,351,496,811,423,729,200,603
,555,573,694,613,29,205,210,6550
160 DATA 126,198,962,783,491,30,155,14
9,165,664,922,282,984,546,128,6585
1060 DATA 935,150,610,902,831,393,206,
176,260,139,808,794,46,226,882,7358
1210 DATA 131,146,200,935,5,841,50,688
,71,755,994,67,182,274,571,5910
1360 DATA 914,577,146,147,630,600,957,
868,3,373,938,541,827,230,251,8002
1510 DATA 412,913,920,795,644,634,908,
931,136,99,182,703,906,416,526,9125
1660 DATA 468,727,71,30,556,46,366,116
,32,46,153,69,19,939,726,4364
1810 DATA 703,590,358,106,257,888,174,
243,783,663,881,961,979,76,938,8600
1960 DATA 986,316,44,853,937,937,837,8
72,993,559,434,731,938,648,10085
```

•

```
*****************************
*                           *
*        BACTERION!         *
*     THE PLAGUE OF 2369    *
*           BY              *
*       KYLE PEACOCK        *
*          WITH             *
*       TOM HUDSON          *
*                           *
* ANALOG COMPUTING MAGAZINE *
*   ALL RIGHTS RESERVED     *
*****************************

* ATARI MEMORY USAGE

CASINI  EQU $0002 ;RESTART VECTOR
DOSVEC  EQU $000A ;RESTART VECTOR
DOSINI  EQU $000C ;RESTART VECTOR
SDMCTL  EQU $022F ;DMA ENABLE SHADOW
PMBASE  EQU $D407 ;PM BASE REG.
GRACTL  EQU $D01D ;GRAPHICS CONTROL
GPRIOR  EQU $026F ;PRIORITY CONTROL
AUDC1   EQU $D201 ;AUDIO CHANNELS
AUDC2   EQU $D203
AUDC3   EQU $D205
AUDC4   EQU $D207
AUDF1   EQU $D200 ;AUDIO FREQUENCIES
AUDF2   EQU $D202
AUDF3   EQU $D204
AUDF4   EQU $D206
AUDCTL  EQU $D208 ;AUDIO CONTROL
HITCLR  EQU $D01E ;CLEAR COLLISIONS
WSYNC   EQU $D40A ;WAIT FOR HOR SYNC.
RANDOM  EQU $D20A ;RANDOM #s
PCOLR0  EQU $02C0 ;COLOR OF PLR/MIS
COLPF0  EQU $D016 ;PLAYFIELD COLOR #0
COLOR0  EQU $02C4 ;COLOR REGISTERS
COLOR4  EQU $02C8 ;BACKGROUND COLOR
NMIEN   EQU $D40E ;NMI INTER. ENABLE
CH      EQU $02FC ;LAST KEY PRESSED
SKSTAT  EQU $D20F ;KEYBOARD STATUS
RTCLOK  EQU $12   ;INTERNAL CLOCK
VDSLST  EQU $0200 ;DLI VECTOR
SDLSTL  EQU $0230 ;DLIST VECTOR
HPOSP0  EQU $D000 ;HOR. POS. PLR 0
HPOSP1  EQU $D001 ;HOR. POS. PLR 1
HPOSP2  EQU $D002 ;HOR. POS. PLR 2
HPOSP3  EQU $D003 ;HOR. POS. PLR 3
HPOSM0  EQU $D004 ;HOR. POS. MISS 0
HPOSM1  EQU $D005 ;HOR. POS. MISS 1
HPOSM2  EQU $D006 ;HOR. POS. MISS 2
HPOSM3  EQU $D007 ;HOR. POS. MISS 3
ATRACT  EQU $4D   ;ATTRACT MODE FLAG
CONSOL  EQU $D01F ;CONSOLE BUTTONS
STICK0  EQU $0278 ;JOYSTICK PORT 0
P0PL    EQU $D00C ;PLR/PLR COLLISIONS
M0PL    EQU $D008 ;MIS/PLR COLLISIONS
P0PF    EQU $D004 ;PLR/PLF COLLISIONS
TRIG0   EQU $D010 ;JOYSTICK BUTTON 0

* ATARI HARDWARE REGISTERS

SETVBV  EQU $E45C ;SET SYS. TIMERS
SYSVBV  EQU $E45F ;1st STAGE VBLANK
XITVBV  EQU $E462 ;X-IT VBLANK
SIOINV  EQU $E465 ;SIO INIT

* PLAYER/MISSILE DATA AREA

PLAYBS  EQU     $0000
MISS    EQU     PLAYBS+0768
PLAY0   EQU     PLAYBS+1024
PLAY1   EQU     PLAYBS+1280
PLAY2   EQU     PLAYBS+1536
PLAY3   EQU     PLAYBS+1792

* DISPLAY DATA AREA

DLIST   EQU     $1F00
DISP    EQU     $2010
DISP2   EQU     $3000

* ZERO PAGE VARIABLES

        ORG     $80

LO      DS 1
HI      DS 1
BALO    DS 1
BAHI    DS 1
DRWLO   DS 1
DRWHI   DS 1
PLLO    DS 1
PLHI    DS 1

*       VARIABLES

* PRIMARY USE IN 'PLOTTER' ROUTINE.

PLOTX   DS 1 ;PLOT X-COORD.
PLOTY   DS 1 ;PLOT Y-COORD.
COLOR   DS 1 ;POINT COLOR.
DRAWX   DS 1 ;DRAWTO X-COORD.
DRAWY   DS 1 ;DRAWTO Y-COORD.
ACCX    DS 1 ;X ACCUM.
ACCY    DS 1 ;Y ACCUM.
DELTAX  DS 1 ;DRAW WORK AREA.
DELTAY  DS 1 ;DRAW WORK AREA.
INCX    DS 1 ;DRAW X INCREMENT.
INCY    DS 1 ;DRAW Y INCREMENT.
COUNTR  DS 1 ;DRAWTO COUNTER.
ENDPT   DS 1 ;DRAWTO ENDPOINT.
HOLD    DS 1 ;WORK AREA.
XWORK   DS 1
YWORK   DS 1
YOFSET  DS 1 ;PLOT Y OFFSET.
SHAPIX  DS 1 ;OBJECT #.
SHAPCT  DS 1 ;OBJECT SHAPE COUNTER.
XI      DS 1 ;OBJECT X INCREMENT.
YI      DS 1 ;OBJECT Y INCREMENT.
LENGTH  DS 1 ;OBJECT (TAKE A GUESS!)

* PRIMARY USE IN 'TEST' ROUTINE.

VSTOP   DS 1  ;STOP VBLANK FLAG.
LSOUND  DS 1  ;LASER SOUND FLAG.
XSOUND  DS 1  ;DETONATION SOUND FLAG.
CSOUND  DS 1  ;CANNON SOUND FLAG.
FREQ    DS 1  ;PULSE SOUND FREQUENCY.
DEMO    DS 1  ;GAME UNDERWAY FLAG.
STRUCT  DS 1  ;CELL STRUCTURE #.
NOPLAY  DS 1  ;# OF PLAYERS.
SDELAY  DS 2  ;PULSE SOUND DELAY.
LISTPT  DS 1  ;DISPLAY LIST POINTER.
SCORE1  DS 2  ;LO-BYTE OF SCORES.
SCORE2  DS 2  ;MD-BYTE OF SCORES.
SCORE3  DS 2  ;HI-BYTE OF SCORES.
TSCR1   DS 3  ;TOTAL SCORE BYTES.
CELNUM  DS 1  ;CELL # BEING DRAWN.
CELREF  DS 1  ;CELL # BEING REFRESHED

* PRIMARY USE IN 'STRAT' ROUTINE.

GEVNUM  DS 1  ;ATTACKING BACTERION! #
FRANGE  DS 1  ;ATTACK PLAYER RADIUS.
GEVESC  DS 1  ;# OF ESCAPED VESSELS.
TOTCEL  DS 1  ;# OF CELLS REMAINING.
TURNT   DS 1  ;BACTERION! TURN TIMERS
FIRETM  DS 1  ;RAM COPY OF 'GEVFRE'

* PRIMARY USE IN 'SHOOT' ROUTINE

FDELAY  DS 2  ;DELAY BETWEEN SHOTS.
NOBULL  DS 2  ;# OF BULLETS FIRED.
NOKILL  DS 2  ;# OF VESSELS KILLED.

* PRIMARY USE IN 'TEST' ROUTINE

CELLMV  DS 20 ;CELL MOVING FLAGs.
CELLNX  DS 20 ;CELL NEW X-COORD.
CELLOX  DS 20 ;CELL OLD X-COORD.

        ORG     $1CB5

CELLNY  DS 20 ;CELL NEW Y-COORD.
CELLOY  DS 20 ;CELL OLD Y-COORD.
GEVCEL  DS 20 ;CELL # BEING HEISTED.

* PRIMARY USE IN 'STRAT' ROUTINE

GEVFRE  DS 20 ;BACTERION! FIRE TIMERS
MOVET   DS 20 ;BACTERION! MOVE TIMERS
GEVDES  DS 20 ;DESIRED DIRECTION.
GEVDIR  DS 20 ;ACTUAL DIRECTION.
ATTACK  DS 20 ;ATTACKING PLAYER FLAG.
ESCAPE  DS 20 ;Y-COORD FOR ESCAPING.
STOP    DS 20 ;BACTERION! ICED FLAG.
GEVX    DS 20 ;BACTERION! X-COORD.
GEVY    DS 20 ;BACTERION! Y-COORD.
TARX    DS 20 ;BACTERION! TARGET-X.
TARY    DS 20 ;BACTERION! TARGET-Y.
LSRDIR  DS 20 ;LASER FIRING DIRECTION
LSRTME  DS 20 ;LASER LIFE TIMER.
LASERX  DS 20 ;LASER X-COORD.
LASERY  DS 20 ;LASER Y-COORD.

* PRIMARY USE IN 'DRAW' ROUTINE.

TYPE    DS 20 ;VESSEL TYPE.
PHASE   DS 20 ;VESSEL PHASE.
TYPES   DS 1  ;ATTACKING TYPE.

* PRIMARY USE IN 'SHOOT' ROUTINE.

SPEED   DS 20 ;SPEED A PLAYER MOVES.
CSPEED  DS 20 ;RAM COPY OF 'SPEED'
BULLET  DS 20 ;BULLET OWNER (0 OR 1)

 TITLE 'BACTERION! MASTER ASSEMBLY'

        INCLUDE D:GEV.TXT

        ORG $2800
        LOC $0000
 TITLE 'ONE SHOT INITIALIZER'

        PROC

*       RELOCATE CODE

MOVEIT  JSR SIOINV      ;INIT SOUNDS.
        LDA #$22        ;INITIALIZE
        STA SDMCTL      ;DMA CONTROL.
        LDA #HIGH $2800 ;SET ORIGIN
        STA HI          ;ADDR. IN
        LDA #LOW $2800  ;2-BYTE
        STA LO          ;POINTER.
        LDA #HIGH $0800 ;SET DESTIN-
        STA BAHI        ;ATION IN 2-
        LDA #LOW $0800  ;BYTE
        STA BALO        ;POINTER.
        LDY #$00        ;RESET Y-REG.
MOVELP  LDA (LO),Y      ;MOVE A 256
        STA (BALO),Y    ;BYTE BLOCK
        INY             ;USING Y-REG.
        BNE MOVELP      ;
        INC HI          ;NEXT 256-
        INC BAHI        ;BYTE BLOCK.
        LDA HI          ;
        CMP #$40        ;DONE YET?
        BNE MOVELP      ;NO-KEEPITUP.
        LDA #HIGH TEST  ;SNAG ALL
        STA DOSINI+1    ;RESET
        STA CASINI+1    ;VECTORS
        STA DOSVEC+1    ;SO AS TO
        LDA #LOW TEST   ;GAIN
        STA DOSINI      ;COMPLETE
        STA CASINI      ;CONTROL OF
        STA DOSVEC      ;THE SYSTEM.
        JMP TEST        ;ALL DONE. START!

*       BUILD GAME BOARD DISPLAY LIST

INIT    LDX #$00        ;RESET X-REG.
        LDA #$0E        ;DLIST OPCODE
BDLOOP  STA DLIST,X     ;STORE IT.
        DEX             ;DONE YET?
        BNE BDLOOP      ;NO! GO BACK!
        LDA #$70        ;
        STA DLIST+0     ;INSTALL
        STA DLIST+1     ;REMAINDER OF
        LDA #$F0        ;SPECIAL
        STA DLIST+2     ;DISPLAY
        LDA #$4E        ;LIST OP-
        STA DLIST+3     ;CODES & OP-
        STA DLIST+107   ;ERANDS INTO
        LDA #HIGH DISP  ;DISPLAY
        STA DLIST+5     ;LIST.
        LDA #LOW DISP   ;
        STA DLIST+4     ;
        LDA #HIGH DISP2 ;
        STA DLIST+109   ;
        LDA #LOW DISP2  ;
        STA DLIST+108   ;
        LDA #$41        ;
        STA DLIST+200   ;
        LDA #$8E        ;
        STA DLIST+199   ;
        LDA #HIGH DLIST ;
        STA DLIST+202   ;
        LDA #LOW DLIST  ;
        STA DLIST+201   ;

*       PLAYER/MISSILE INITIALIZATION

        LDA #$3E        ;
        STA SDMCTL      ;DMA ENABLE.
        LDA #HIGH PLAYBS ;PM BASE
        STA PMBASE      ;ADDRESS.
        LDA #$03        ;GRAPHICS
        STA GRACTL      ;CONTROL.
        LDA #$10        ;PRIORITY
        STA GPRIOR      ;REGISTERS.
        LDA #$00        ;CLEAR OUT
        TAX             ;PLAYER 0
:PMSET  STA PLAY0,X     ;& PLAYER 1
        STA PLAY1,X     ;
        DEX             ;ALL DONE?
        BNE :PMSET      ;NO! CONT.

*       CLEAR OUT PLAYERS' SCORES

        LDX #$02
        LDA #$00        ;FILL WITH 0
:CLRS2  STA SCORE1,X
        STA SCORE2+1,X
        STA TSCR1,X
        DEX             ;ALL DONE?
        BPL :CLRS2      ;NO! CONTINUE

*       CLEAR PLAYFIELD AREA

        LDA #LOW DISP   ;I'LL LET
        STA LO          ;YOU GUYS
        LDA #HIGH DISP  ;FIGURE OUT
        STA HI          ;WHAT'S GOING
        LDX #0          ;ON HERE!
CDLP    LDA LO
        STA LOTBL,X     ;I'M SO LAZY!
        LDA HI
        STA HITBL,X
        LDA #0
        LDY #39
CDLP2   STA (LO),Y
        DEY
        BPL CDLP2
        INX
        CPX #192
        BEQ DOIT
        LDA LO
        CLC
        ADC #40
        STA LO
        LDA HI
        ADC #0
        STA HI
        JMP CDLP

*       SET UP ATOMIC PILE CELLS

DOIT    LDX STRUCT
        LDY STRBSE,X
        LDX #9
        STX TOTCEL
SETCEL  LDA :ICELX,Y ;X-COORDS
        STA CELLOX,X
        STA CELLNX,X
        LDA :ICELY,Y ;Y-COORDS
        STA CELLOY,X
        STA CELLNY,X
        LDA #$00
        STA CELLMV,X ;CELLS NOT MOV-
        INY          ;ING STATUS.
        DEX
        BPL SETCEL

*       SET UP PLAYERS PROBEs

        LDX #$00    ;FLIP TO GAME
        STX LISTPT  ;BOARD SCREEN.
        STX GEVX+0  ;X-COORD.
        STX GEVX+1  ;X-COORD.
        JSR SETPLR  ;SET IT UP NOW.
        LDA NOPLAY  ;1 PLAYER GAME?
        BEQ SETTYP  ;YES! SKIP #2
        INX         ;NO! SET UP PLR 2.
        JSR SETPLR  ;SET IT UP NOW.

*       ATTACKING BACTERION! TYPE

SETTYP  LDA #$01    ;SLOWEST TYPE
        STA TYPES   ;ATTACKS FIRST.
        STA TYPE+2
        STA TYPE+3
        STA TYPE+4

* BACTERION! INITIAL A.I. SPEEDS

        LDX #$05
SETMDB  LDA IMOVDB,X ;MOVEMENT SPEED
        STA MOVEDB,X
        LDA ITURDB,X ;TURN SPEED
        STA TURNDB,X
        LDA IFIRDB,X ;FIRE SPEED
        STA FREBSE,X
        DEX
        BPL SETMDB

        JMP INIT2

*       INITIALIZE GIVEN PLAYER

SETPLR  LDA #$00      ;PLAYER NO LONG-
        STA STOP,X    ;ER DEAD OR EXP-
        STA TYPE,X    ;LODING.
        LDA #$01      ;SET UP FOR
        STA SPEED,X   ;COASTING SPEED.
        STA CSPEED,X  ;
        LDA PLRPHS,X  ;CORRECT TYPE
        STA PHASE,X   ;OF VESSEL
        STA GEVDIR,X  ;PHASE.
        LDA PLRX,X    ;PROPER X-COORD.
        STA GEVX,X    ;
        LDA #128      ;PROPER Y-COORD.
        STA GEVY,X    ;
        RTS           ;ALL DONE...

PLRX    DB 44,195 ;INITIAL X-COORDS
PLRPHS  DB 12,4   ;INITIAL DIRECTIONS

* BACTERION! INITIAL TURNING DATABASE

ITURDB  DB 52,44,36,28,20,12

* BACTERION! INITIAL MOVING DATABASE

IMOVDB  DB 13,11,09,07,05,03

* BACTERION! INITIAL FIRING TIMES

IFIRDB  DB 70,65,60,55,50,45

* OFFSET DATABASE TO CELL FORMATIONS

STRBSE  DB 0,10,20

* CELLS INITIAL X-COORDS
```

```
:ICELX  DB 68,80,92,62,74
        DB 86,98,68,80,92

        DB 79,79,79,79,73
        DB 85,67,91,61,97

        DB 79,71,87,79,79
        DB 79,79,71,87,79

* CELLS INITIAL Y-COORDS

:ICELY  DB 93,93,93,96,96
        DB 96,96,99,99,99

        DB 76,82,88,94,98
        DB 98,102,102,106,106

        DB 86,90,90,94,94
        DB 94,94,98,98,102
 TITLE 'BACTERION! MULTI-INITIALIZER'

INIT2   PROC

*       ALLOW THINGS TO SETTLE DOWN

        LDX #$01  ;STOP VERTICAL
        STX VSTOP ;BLANK ROUTINE.

*       INITIALIZE SOUNDS

        DEX       ;
        TXA       ;
        STA AUDC3 ;TURN OFF SOUND
        STA AUDC4 ;REGISTERS & INIT
        STA AUDCTL ;SOUND CHANNELS.

*       CLEAR PLAYER MISSILE AREA

:ER1    STA MISS,X  ;BACTERION! #1.
        STA PLAY2,X ;BACTERION! #2.
        STA PLAY3,X ;BACTERION! #3.
        STA HITCLR  ;CLEAR COLLISIONS
        DEX
        BNE :ER1

* ENEMY BACTERION!s X,Y,& DIRECTION

        LDX #$04    ;HANDLE ALL.
SETGEV  LDA TYPES   ;GET TYPE OF ATT-
        STA TYPE,X  ;ACKING BACTERION.
        JSR PICKX   ;RANDOM X-COORD.
        STA GEVX,X  ;STORE IT.
        LDA RANDOM  ;RANDOM NUMBER.
        BPL :PLUS1  ;BRANCH IF > 0.
        LDA #21     ;INITIAL Y-COORD.
        BNE :PLUS2  ;BRANCH!
:PLUS1  LDA #228    ;INITIAL Y-COORD.
:PLUS2  STA GEVY,X  ;STORE IT.
        JSR PICKDR  ;RANDOM DIRECTION.

* ENEMY BACTERION!Rs TARGET CELL

SEL     LDA #$00    ;CLEAR
        STA TRY     ;COUNTERS,
        STA GEVESC  ;# OF ESCAPEES,
        STA STOP,X  ;DEATH STATUS,
        STA GEVCEL,X ;TARGET CELL,
        STA LSRDIR,X ;LASER FLAG,
        STA GEVFRE,X ;FIRE TIME FLAG,
        STA ESCAPE,X ;ESCAPE Y-COORD.

        LDA RANDO ;GET RANDOM #
        CMP #$D8  ;GREATER THAN $D8
        BCS SELIT ;YES, BRANCH!

SEL0    LDA #$0A  ;RANDOM # SEED
        JSR RANDO ;GET RANDOM CELL #
        TAY       ;STORE IN Y-REG.

        STX :XHOLD   ;SAVE X-REG.
CPICK0  LDA ESCAPE,X ;ESCAPE Y-COORD
        BEQ CPICK1   ;EQUAL 0?
        TYA          ;NO! MOVE Y TO A
        CMP GEVCEL,X ;IS THIS CELL
        BEQ SEL1     ;SPOKEN FOR?
CPICK1  INX          ;NO! MOVE ON
        CPX #$05     ;TO NEXT
        BNE CPICK0   ;BACTERION!
        LDX :XHOLD   ;RESTORE X-REG.

*       HEIST THIS CELL (IN Y-REG.)

        LDA CELLNY,Y ;CELL Y-COORD.
        CMP #194     ;OFF SCREEN?
        BCS SEL1     ;YES! TRY AGAIN.
        LDA #$02     ;RANDOM # SEED.
        JSR RANDO    ;GET RANDOM #.
        STY :YHOLD   ;SAVE Y-REG.
        TAY          ;MOVE # TO ACC.
        LDA ESCDT,Y  ;GET PROPER ES-
        STA ESCAPE,X ;CAPE Y-COORD.
        LDA CELDT,Y  ;GET OFFSET FOR
        LDY :YHOLD   ;LOCKING ON TO
        CLC          ;TOP OR BOTTOM
        ADC CELLNY,Y ;OF CELL & STORE
        STA TARY,X   ;IN TARGET-Y.
        LDA CELLNX,Y ;GET X-COORD OF
        CLC          ;CELL & STORE
        ADC #44      ;IN TARGET-X.
        STA TARX,X   ;
        TYA          ;MAKE THIS CELL
        STA GEVCEL,X ;SPOKEN FOR.
        JMP SEL2     ;HANDLE NEXT.

SEL1    LDX :XHOLD   ;RESTORE X-REG.
        DEC TRY      ;DEC. COUNTERS
        BNE SEL0     ;& BRANCH!

* CAN'T FIND A CELL, ATTACK PLAYERs

SELIT   LDY NOPLAY ;# OF PLAYERS.
        INY        ;ADD ONE.
        TYA        ;RANDOM # SEED.
        JSR RANDO  ;GET RANDOM #.
        TAY        ;MOVE TO Y-REG.
        LDA GEVX,Y ;PLAYER'S X-COORD.
        STA TARX,X ;USE AS TARGET-X.
        LDA GEVY,Y ;PLAYER'S Y-COORD.
        STA TARY,X ;USE AS TARGET-Y.

SEL2    DEX        ;HANDLE NEXT
        CPX #$01   ;BACTERION! (IF
        BEQ :NEXT  ;WE AREN'T DONE.)
        JMP SETGEV ;JUMP TO IT!

*       SET UP RANGE OF ATTACK

:NEXT   LDA #40
        STA FRANGE

*       SET UP FIRING TIME

        LDX TYPES       ;ATTACK TYPE.
        LDA FREBSE-1,X  ;GET DATA FROM
        STA FIRETM      ;FIRING DB.
        CMP #$03        ;IS IT < 5?
        BCC PROJT1      ;NO! BRANCH!
        SEC             ;YES! SUBTRACT
        SBC #$03        ;3 & STORE
        STA FREBSE-1,X  ;IT IN DB.

*       CLEAR PROJECTILE WORK AREA

PROJT1  LDX #12         ;HANDLE ALL.
        LDA #$FF        ;INACTIVE STATUS.
CLRPRO  STA BULLET,X    ;OWNER.
        STA CELLNX+5,X  ;X-COORD.
        STA CELLNY+5,X  ;Y-COORD.
        STA CELLMV+5,X  ;MOVING.
        DEX             ;HANDLE NEXT
        CPX #$04        ;(IF ANY).
        BNE CLRPRO      ;

*       SET UP COLOR DATABASES

        LDX #$03    ;COUNTER.
:SET1   LDA RANDOM  ;RANDOM #.
        AND #$F0    ;ZAP LO-NIBBLE.
        ORA #$0A    ;& W/DECINAL 10.
        STA PCOLR0,X ;STORE IT.
        LDA RANDOM  ;RANDOM #.
        AND #$F0    ;ZAP LO-NIBBLE.
        ORA :FIELD,X ;OR W/PLAYFIELD.
        STA COLOR0,X ;STORE IT.
        DEX         ;HANDLE NEXT.
        BPL :SET1   ;BRANCH!

        LDA #$00    ;CLEAR.
        STA NOBULL+0 ;# OF BULLETS
        STA NOKILL+0 ;FIRED & # OF
        STA NOBULL+1 ;BACTERIONS!
        STA NOKILL+1 ;VAPORIZED.

        STA VSTOP   ;START VBLANK.

        RTS         ;BUG OFF!!!

* BACTERION! FIRE TIME DATABASE

FREBSE  DB 0,0,0,0,0,0

* OFFSETS TO TOP & BOTTOM OF CELL

CELDT   DB 21,36

* Y-COORDS FOR ESCAPING BACTERION!

ESCDT   DB 10,245

* COLOR LUM. FOR PLAYERS & PLAYFIELDS

:PLAYC  DB $0C,$0C,$0A,$0A
:FIELD  DB $0C,$08,$08,$0A

* PICK A RANDOM X-COORD.

PICKX   LDA #100
        JSR RANDO
        CLC
        ADC #75
        RTS

* PICK A RANDOM DIRECTION (0-15)

PICKDR  LDA #16
        JSR RANDO
        STA GEVDIR,X
        RTS

* PICK A RANDOM # (0 UP TO ACC.)

RANDO   STA HOLDME
        LDA RANDOM
RANDO1  CMP HOLDME
        BCC RANOUT
        LSR A
        LSR A
        JMP RANDO1
RANOUT  RTS

HOLDME  DB 0 ;TEMP STORAGE.
TRY     DB 0 ;COUNTER.
:XHOLD  DB 0 ;X-REG TEMP STORAGE.
:YHOLD  DB 0 ;Y-REG TEMP STORAGE.
 TITLE 'GET THE GAME GOING...'

TEST    PROC

        LDA #$00    ;TURN OFF
        STA AUDC1   ;SOUND
        STA AUDC2   ;REGISTERS.

        JSR :SETP0  ;SET UP TITLE
        JSR ADITUP  ;SCREEN SCORES
        LDA #$01    ;& SHOW TITLE
        STA LISTPT  ;SCREEN.
        STA VSTOP   ;

        LDA #$06    ;SET UP VERT-
        LDX #HIGH VBL ;ICAL BLANK
        LDY #LOW VBL ;ROUTINES.
        JSR SETVBV  ;

        LDA #$07    ;SET UP DEF.
        LDX #HIGH DBL ;VERTICAL
        LDY #LOW DBL ;BLANK
        JSR SETVBV  ;ROUTINES.

        LDA #$C0    ;SET UP DLI
        STA NMIEN   ;ROUTINES.

:DEMOX  JSR SHUTUP
        LDA DEMO    ;WAIT FOR PUSH
        CMP #$01    ;OF START KEY.
        BNE :DEMOX  ;
        INC DEMO    ;

        JSR INIT    ;INITIALIZE...

*       START OF NEW GAME

SAVE0   LDA DEMO    ;CHANGE STATUS
        CMP #$02    ;OF 'DEMO'
        BNE TEST    ;VARIABLE

*       GAME PAUSED?

        LDA CH      ;IS SPACEBAR
        CMP #$21    ;PRESSED?
        BNE :REFCL  ;YES. IS BAR
        LDA SKSTAT  ;STILL BEING
        AND #$04    ;PRESSED?
        BEQ :REFCL  ;YES-CONTINUE.

        JSR ADITUP  ;ADD UP SCORES

        LDA TOTCEL  ;HOLD # OF
        PHA         ;CELLS LEFT.
        LDA #$0A    ;MAKE WEIRD
        STA TOTCEL  ;SOUND.

        LDA #$01    ;SWITCH TO
        STA LISTPT  ;TITLE SCREEN.
        STA VSTOP
        STA CH      ;WAIT FOR
        JSR SHUTUP  ;SPACEBAR.

:WAIT1  LDA CH      ;IS SPACEBAR
        CMP #$21    ;PRESSED? NO,
        BNE :WAIT1  ;SO WAIT.
:WAIT2  LDA SKSTAT  ;YES. IS BAR
        AND #$04    ;STILL BEING
        BEQ :WAIT2  ;PRESSED?

        PLA         ;RESTORE # OF
        STA TOTCEL  ;CELLS LEFT.

        LDA #$00    ;SWITCH BACK
        STA LISTPT  ;TO PLAYFIELD
        STA VSTOP   ;& CONTINUE...
        STA CH

*   REFRESH CELLS (ONE PER PASS)

:REFCL  DEC CELREF  ;OUT OF CELLS
        BPL GETCEL  ;TO REFRESH?
        LDA #$09    ;YES. START
        STA CELREF  ;AGAIN...
GETCEL  LDX CELREF  ;GET CELL # TO
        LDA CELLMV,X ;REFRESH & SEE
        BNE GOTCEL  ;IF IT MOVES.

        LDA CELLOX,X ;CELL X-COORD
        STA PLOTX   ;
        LDA CELLOY,X ;CELL Y-COORD
        STA PLOTY   ;
        LDA #1      ;SPEC. COLOR
        STA COLOR   ;
        LDA #0      ;SPEC. OBJECT
        JSR OBJECT  ;DRAW IT...

*   DRAW MOVING CELLS (ALL AT ONCE)

GOTCEL  LDA #$09    ;SET UP CELL #
        STA CELNUM  ;TO DRAW.
SAVE1   LDX CELNUM  ;GET CELL #.
        LDA CELLMV,X ;IS IT BEING
        BEQ SAVE2   ;CARRIED OFF?
        JSR SHOCEL  ;YES. DRAW IT.
SAVE2   DEC CELNUM  ;MOVE ON TO
        BPL SAVE1   ;NEXT CELL.

*   DRAW & ERASE BACTERION! LASERS

        LDA #$04    ;CHECK ALL
        STA LSRCNT  ;VESSELS.
LASER3  LDX LSRCNT  ;IS THIS VES-
        LDA LSRDIR,X ;SEL FIRING?
        BEQ NXTLSR  ;NO. CONTINUE.
        BMI LASER5  ;YES. ERASE?
        ORA #$80    ;SET UP LASER
        STA LSRDIR,X ;TO BE ERASED.
        LDA #$02    ;SPECIFY COLOR
        STA COLOR   ;OF LASER.
        LDA GEVX,X  ;GET X-COORD
        SEC         ;OF FIRING
        SBC #44     ;VESSEL & USE
        STA LASERX,X ;AS LASER
        STA PLOTX   ;X-COORD.
        LDA GEVY,X  ;GET Y-COORD
        SEC         ;OF FIRING
        SBC #$1C    ;VESSEL & USE
        CMP #192    ;AS LASER
        BCS LASER6  ;Y-CORD. DO
        STA LASERY,X ;NOT FIRE IF
        STA PLOTY   ;NOT ON-SCREEN
        LDA LSRDIR,X ;ELSE, DRAW
        AND #$7F    ;FIRING LASER
        JSR OBJECT  ;(DEATH RAY).
        JMP NXTLSR  ;HANDLE NEXT.
LASER5  DEC LSRTME,X ;TIME TO
        BPL NXTLSR  ;ERASE LASER?
        LDA LASERX,X ;YES. GET X
        STA PLOTX   ;COORD & Y
        LDA LASERY,X ;COORD FOR
        STA PLOTY   ;ERASING.
        LDA #$00    ;SPECIFY COLOR
        STA COLOR   ;AS BCKGROUND.
        LDA LSRDIR,X ;NOW ERASE
        AND #$7F    ;LASER.
        JSR OBJECT  ;
        LDX LSRCNT  ;TURN OFF
LASER6  LDA #$00    ;LASER FOR
        STA LSRDIR,X ;THIS VESSEL.
NXTLSR  DEC LSRCNT  ;HANDLED ALL
        LDA LSRCNT  ;LASERS? IF
        CMP #$01    ;SO, QUIT.
        BNE LASER3  ;ELSE GO BACK.

*       DRAW & ERASE PROJECTILES

        LDX #17     ;HANDLE ALL.
PROJ5   LDA CELLMV,X ;IS THIS BULLET
        BEQ PROJ6   ;ACTIVE?
        LDA CELLOX,X ;YES. GET X
        STA PLOTX   ;COORD.
        LDA CELLOY,X ;GET Y-COORD.
        STA PLOTY   ;OF BULLET.
        LDA #$00    ;SET UP TO ERASE
        STA CELLMV,X ;THIS BULLET.
        STA COLOR   ;SPECIFY COLOR.
        STX :XHOLD  ;SAVE X-REG.
        JSR PLOTPT  ;ERASE IT NOW!
        LDX :XHOLD  ;RESTORE X-REG.
        LDA CELLNX,X ;GET NEW BULLET
        STA CELLOX,X ;X-COORD & PRE-
        STA PLOTX   ;PARE TO PLOT.
        LDA CELLNY,X ;GET NEW BULLET
        STA CELLOY,X ;Y-COORD & PRE-
        STA PLOTY   ;PARE TO PLOT.
        LDA #$03    ;SPECIFY COLOR
        STA COLOR   ;OF NEW BULLET.
        STX :XHOLD  ;SAVE X-REG.
        JSR PLOTPT  ;DRAW NEW BULLET
        LDX :XHOLD  ;RESTORE X-REG.
PROJ6   DEX         ;MOVE ON TO NEXT
        CPX #$09    ;BULLET. IF NONE
        BNE PROJ5   ;LEFT, QUIT.

*       ALL CELLS GONE?
```

```
GAME3   LDA TOTCEL    ;TOTAL # OF
        BPL GAME1     ;CELLS < 0?
        JSR SHUTUP    ;YES. SOUND OFF.
        LDA #$01      ;STOP ALL SCREEN
        STA VSTOP     ;ACTION.
        STA RTCLOK+2  ;SET UP TO
        LDX RTCLOK+1  ;PAUSE FOR
        INX           ;SOME TIME.
HALT    CPX RTCLOK+1  ;IS PAUSE DONE?
        BNE HALT      ;NO. WAIT.
        JMP TEST      ;YES. RESTART.

*BACTERION!s DESTROYED OR OFF SCREEN?

GAME1   LDA GEVESC    ;IS NUMBER OF
        CMP #$03      ;ESCAPED VESSELS
        BEQ GAMEE     ;EQUAL TO THREE?
        JMP SAVE0     ;NO, GO BACK.

GAMEE   JSR ADITUP    ;ADD UP SCORES.

        INC TYPES     ;SET UP NEXT TYPE
        LDA TYPES     ;OF ATTACKING
        CMP #$07      ;BACTERION.
        BCC GAME2     ;

        LDA RANDOM    ;GET RANDOM
        AND #$F0      ;BACKGROUND
        STA COLOR4    ;COLOR.

*       FLIP TO TITLE SCREEN

        LDA #$01      ;STOP ALL SCREEN
        STA VSTOP     ;ACTION & FLIP
        STA LISTPT    ;TO TITLE SCREEN

        JSR SHUTUP    ;SOUNDS OFF.
        LDA TOTCEL    ;SAVE TOTAL #
        PHA           ;OF CELLS LEFT.
        LDA #$0A      ;MAKE WEIRD
        STA TOTCEL    ;TITLE SOUND.

        STA RTCLOK+2  ;PAUSE FOR A
        LDX RTCLOK+1  ;WHILE.
        INX           ;
:PAUSE  CPX RTCLOK+1  ;PAUSE TIME UP?
        BNE :PAUSE    ;NO. WAIT.

        PLA           ;RESTORE TOTAL #
        STA TOTCEL    ;OF CELLS LEFT.

        LDA #$00      ;FLIP TO GAME
        STA LISTPT    ;SCREEN & BEGIN
        STA VSTOP     ;SCREEN ACTION.
        STA CH
        STA XSOUND
        STA CSOUND
        STA LSOUND

* SPEED UP BACTERION! MOVE/TURN RATES

        LDX #$05      ;UPDATE ALL.
FAST1   LDA MOVEDB,X  ;IS MOVE RATE AT
        CMP #$02      ;LOWEST LEVEL?
        BCC FAST2     ;YES, CONTINUE.
        DEC MOVEDB,X  ;NO. DECREMENT.
FAST2   LDA TURNDB,X  ;IS TURN RATE AT
        CMP #08       ;LOWEST LEVEL?
        BCC FAST3     ;YES, CONTINUE.
        SEC           ;NO. DECREMENT.
        SBC #$04      ;
        STA TURNDB,X  ;
FAST3   DEX           ;UPDATE NEXT.
        BPL FAST1     ;

        LDA #$01      ;MAKE ATTACKING
GAME2   STA TYPES     ;BACTERION! THE
        STA TYPE+2    ;SLOWEST TYPE
        STA TYPE+3    ;FOR BEGINNING
        STA TYPE+4    ;OF NEW WAVE.

*       SET UP FOR NEXT ATTACK

        JSR INIT2     ;INITIALIZE...
        JMP SAVE0     ;BEGIN AGAIN.

*       VERTICAL BLANK ROUTINE

VBL     CLD           ;CLEAR DECIMAL
        LDA LISTPT    ;RESEED
        ASL A         ;DISPLAY LIST
        TAX           ;VECTORS ACC-
        LDA :LSTDB+0,X ;ORDING TO
        STA SDLSTL+0  ;'LISTPT'
        LDA :LSTDB+1,X ;VARIABLE.
        STA SDLSTL+1  ;DO THE SAME
        LDA :DLIDB+0,X ;THING FOR
        STA VDSLST+0  ;DLI VECTORS.
        LDA :DLIDB+1,X ;
        STA VDSLST+1  ;

        LDA #$00      ;MAKE ALL PLAYER/
        STA HPOSP0    ;MISSILE X-COORDS
        STA HPOSP1    ;EQUAL TO ZERO.
        STA HPOSP2    ;THIS PROVIDES A
        STA HPOSP3    ;NICE BORDER AT
        STA HPOSM0    ;THE SCREEN TOP.
        STA HPOSM1    ;THE DLIs HANDLE
        STA HPOSM2    ;PROPER PLACEMENT
        STA HPOSM3    ;OF X-COORDS.

        LDA VSTOP     ;IS SCREEN ACTION
        BNE XITVBL    ;HALTED?

*       KILL ATTRACT MODE

        STA ATRACT

*       FLASH CELLS

        LDA COLOR0    ;KEEP SAME LUM.
        CLC           ;BUT UPDATE
        ADC #$10      ;COLOR.
        STA COLOR0    ;

*       BACTERION! ATTACK STRATEGY

        LDA #$04      ;ALL VESSELS
        STA GEVNUM    ;ATTACK!
VBL2    JSR STRAT     ;NASTY ROUTINE!
        DEC GEVNUM    ;NEXT VESSEL
        LDA GEVNUM    ;ATTACKS!
        CMP #$01      ;ALL DONE?
        BNE VBL2      ;IF NOT, CONT.

*       TURN ENEMY BACTERION!s

        JSR TURN

*       MOVE ENEMY BACTERION!s

        JSR MOVE

XITVBL  JMP SYSVBV

*       DEF. VERTICAL BLANK

DBL     CLD           ;CLEAR DECIMAL

*       READ CONSOLE BUTTONS

        JSR BUTTON

*       PULSE SOUND

        JSR PULSE

        LDA VSTOP     ;ALL SCREEN
        BNE XITDBL    ;ACTION HALTED?

*       READ JOYSTICKS & MOVE PLAYERS

        JSR STICKS    ;PLAYERS MOVING
        JSR SHOOT     ;PLAYERS FIRING

*     DON'T LET ANYONE GO OFF SCREEN

        JSR BOUNDS

*     DID ANYBODY RUN INTO ANYBODY?

        JSR COLLIDE

*     KEEP TRACK OF HEISTED CELLS

        JSR TRACK

*       BACTERION! LASER SOUND

        JSR ZAP

*       PLAYER'S CANNON SOUND

        JSR CANNON

*       DETONATION SOUND

        JSR BOOM

*       DRAW PLAYERS & BACTERION!s

        JSR DRAW

XITDBL  JMP XITVBV    ;ALL DONE.

*       PULSE SOUND ROUTINE

PULSE   LDX TOTCEL    ;GET TOTAL # OF
        BMI PULRTS    ;CELLS LEFT.
        LDA FREQS,X   ;GET ASSOCIATED
        STA FREQ      ;PULSE FREQUENCY
        LDA DELS,X    ;& SOUND DELAY
        STA SDELAY+1  ;TIME. STORE IT.
        LDX FREQ      ;STORE PROPER
        STX AUDF1     ;FREQ. INTO
        INX           ;SOUND REGS. 1
        STX AUDF2     ;AND 2.
        LDA SDELAY    ;DECREMENT DELAY
        BEQ SOUND5    ;TIMER IF NON
        DEC SDELAY    ;ZERO.
PULRTS  RTS           ;RETURN.
SOUND5  LDA SDELAY+1  ;RESTORE DELAY
        STA SDELAY    ;TIME IF ZERO.
        INC FLIP      ;TOGGLE PULSE
        LDA FLIP      ;SOUND GENERATOR
        AND #$01      ;(IF OFF, TURN
        TAX           ;ON. IF ON, TURN
        LDA REG,X     ;OFF.) & STORE
        STA AUDC1     ;INTO PROPER
        STA AUDC2     ;AUDIO CHANNEL
        RTS           ;REGISTERS.

REG     DB $00,$A4    ;AUDIO OFF/ON
FLIP    DB 0          ;FLIP/FLOP VAR.
LSRCNT  DB 0          ;LASER COUNTER.

*       PULSE SOUND FREQUENCIES

FREQS   DB 160,170,180,190,200
        DB 210,220,230,240,250,140

*       PULSE SOUND DELAYS

DELS    DB 02,04,06,08,10,12,14
        DB 16,18,20,01

*       BACTERION! LASER SOUND

ZAP     LDX LSOUND    ;IS SOUND ON?
        BEQ ZAPOFF    ;NO. QUIT!
        DEX           ;YES.
        DEC LSOUND    ;GET PROPER
        LDA LFTBL,X   ;FREQ. AND STORE
        STA AUDF3     ;INTO REGISTER.
        LDA LCTBL,X   ;GET PROPER CHAN.
        STA AUDC3     ;AND STORE.
ZAPOFF  RTS           ;ALL DONE...

*       BACTERION! LASER FREQUENCIES

LFTBL   DB 0,236,216,197,177,157
        DB 138,118,99,79,59,40,20,1

*       BACTERION! LASER CHANNELS

LCTBL   DB $00,$A2,$A2,$A2,$A2,$A4
        DB $A4,$A4,$A4,$A4,$A4,$A6
        DB $A6,$A6

*       PLAYER'S CANNON SOUND

CANNON  LDX CSOUND    ;IS SOUND ON?
        BEQ CANOFF    ;NO. QUIT.
        DEX           ;YES. GET PROPER
        DEC CSOUND    ;FREQUENCY &
        LDA CFTBL,X   ;STORE INTO REG.
        STA AUDF4     ;GET PROPER
        LDA CCTBL,X   ;CHANNEL & STORE
        STA AUDC4     ;INTO REG.
CANOFF  RTS           ;ALL DONE...

*       CANNON FREQUENCIES

CFTBL   DB 0,254,212,170,127,85,43,1

*       CANNON CHANNELS

CCTBL   DB $00,$A8,$A8,$A8,$A8,$A8
        DB $A8,$A8

*       DETONATION SOUND

BOOM    LDX XSOUND    ;IS SOUND ON?
        BEQ BOMOFF    ;NO. CONTINUE.
        DEX           ;YES. GET PROPER
        DEC XSOUND    ;FREQ. AND STORE
        LDA XFTBL,X   ;INTO SOUND
        STA AUDF3     ;REGISTER.
        LDA XCTBL,X   ;GET PROPER
        STA AUDC3     ;CHANNEL & STORE.
BOMOFF  RTS           ;ALL DONE...

*       DETONATION FREQUENCIES

XFTBL   DB 0,253,249,245,241,237,234
        DB 230,226,222,218,214,211
        DB 207,203,199,195,191,188
        DB 184,180,176,172,168,165
        DB 161,157,153,149,145,142
        DB 138,134,130,126,123,119
        DB 115,111,107,103,100,96,92
        DB 88,84,80,77,73,69,65,61,57
        DB 54,50,46,42,38,34,31,27,23
        DB 19,15,11,08,04

*       DETONATION CHANNELS

XCTBL   DB $00,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A
        DB $8A,$8A,$8A,$8A,$8A,$8A

*       READ CONSOLE BUTTONS

BUTTON  LDA CONSOL    ;CONSOLE SWITCH.
        CMP #$07      ;IS IT PRESSED?
        BEQ :ACTVE    ;YES, CONTINUE.
        STA :PREV     ;NO, SAVE IT.
        RTS           ;RETURN.
:ACTVE  LDA :PREV     ;GET CONSOLE VALUE
        CMP #$06      ;START BUTTON ON.
        BEQ :START    ;
        CMP #$05      ;SELECT BUTTON ON.
        BEQ :SELCT    ;
        CMP #$03      ;OPTION BUTTON ON.
        BEQ :OPTON    ;
        RTS           ;CONFUSED! RETURN.

*       START KEY PRESSED!!!

:START  LDX #$01      ;GAME START. UP-
        JMP :ENDBT    ;DATE STATUS.

*       SELECT KEY PRESSED!!!

:SELCT  INC NOPLAY    ;UPDATE # OF PLAY-
        LDA NOPLAY    ;ERS. (1 OR 2)
        AND #$01      ;SAVE IN 'NOPLAY'
        STA NOPLAY    ;VARIABLE.

:SETP0  LDX #19       ;CLEAR PLAYERS'
        LDA #00       ;SCORE LINES
:SETP1  STA PLAYR1,X  ;BY PLACING
        STA PLAYR2,X  ;BLANKS IN ALL
        DEX           ;POSITIONS.
        BPL :SETP1    ;

        LDX NOPLAY    ;GET # PLAYERS.
        LDA :PLINE,X  ;DECIDE HOW MANY
        TAX           ;CHARACTERS TO
:SETP2  LDA :LINE1,X  ;USE FOR PLAYER
        STA PLAYR1,X  ;SCORE LINE &
        LDA :LINE2,X  ;INSTALL INTO
        STA PLAYR2,X  ;PLAYER SCORE
        DEX           ;LINE. CONTINUE
        BPL :SETP2    ;UNTIL DONE.
        JMP :ENDBT

:OPTON  INC STRUCT    ;CHANGE STRUCT-
        LDA STRUCT    ;URE OF INITIAL
        CMP #$03      ;CELL PATTERN.
        BCS :OPTON    ;RANGE: 0-2.
        LDX #$00

:ENDBT  STX DEMO      ;UPDATE 'DEMO'
        STX :PREV     ;UPDATE PERVIOUS
        RTS           ;ALL DONE...

:PREV   DB      0     ;OLD CONSOLE VALUE.

:PLINE  DB 10,19      ;SCORE LINE CHARA-
                      ;CTERS PER PLAYER.

*       PLAYER 1/PLAYER 2 SCORE LINE

:LINE1  DB 176,172,161,185,165,178
        DB 00,$D1,00,00,00,00,176
        DB 172,161,185,165,178,00,$D2

*       000000/000000 SCORE LINE

:LINE2  DB 00,$50,$50,$50,$50,$50,$50
        DB 00,00,00,00,00,00
        DB $50,$50,$50,$50,$50,$50,00

*       ADD UP SCORES

ADITUP  SED           ;SET DECIMAL.
        LDX NOPLAY    ;# OF PLAYERS.
CALC4   LDY NOKILL,X  ;# OF KILLS FOR
        BEQ CALCX     ;THIS PLAYER.
CALC5   LDA TYPES     ;BACTERION! TYPE
        ASL A         ;(1 TO 6) IS
        ASL A         ;MULTIPLIED BY
        ASL A         ;10 & ADDED TO
        ASL A         ;PLAYERS' SCORE
        CLC           ;
        ADC SCORE3,X  ;
        STA SCORE3,X  ;
        LDA #$00      ;OVERFLOW
        ADC SCORE2,X  ;CORRECTION.
        STA SCORE2,X  ;
        LDA #$00      ;OVERFLOW
        ADC SCORE1,X  ;CORRECTION.
        STA SCORE1,X  ;
        DEY           ;HANDLED ALL
        BNE CALC5     ;KILLS?
CALCX   DEX           ;HANDLED ALL
        BPL CALC4     ;PLAYERS?

*       DISPLAY SCORES

        LDX #$05      ;SIX DIGITS.
CALC6   LDY NOPLAY    ;# OF PLAYERS.
```

```
        TXA             ;
        AND SCMASK,Y    ;AND IN SCORE
        TAX             ;MASK & GET
        LDA SCORE1,X    ;SCORE DIGIT.
        PHA             ;HOLD IT.
        AND #$0F        ;PLACE '5' IN
        ORA #$50        ;HIGH NIBBLE.
        LDY PUTSCR,X    ;FIND WHERE TO
        STA PLAYR2+1,Y  ;PUT & DO SO.
        PLA             ;RESTORE DIGIT
        LSR A           ;& SHIFT INTO
        LSR A           ;LOW-NIBBLE.
        LSR A           ;
        LSR A           ;PLACE '5' IN
        ORA #$50        ;HIGH-NIBBLE
        STA PLAYR2,Y    ;& STORE.
        DEX             ;IF NOT DONE,
        BPL CALC6       ;CONTINUE.

*       ADD UP TEAM SCORE

        CLC             ;CLEAR CARRY
        LDX #$04        ;LOOP VAR.
        LDY #$02        ;LOOP VAR.
CALC7   LDA SCORE1,X    ;ADD UP CON-
        ADC SCORE1+1,X  ;SECUTIVE
        STA TSCR1,Y     ;SCORES &
        DEX             ;STORE INTO
        DEX             ;TEAM SCORE
        DEY             ;VARIABLES.
        BPL CALC7       ;CONTINUE.

*       DISPLAY TEAM SCORE

        LDX #$02        ;THREE BYTES.
CALC8   LDY PUTTME,X    ;WHERE TO PUT.
        LDA TSCR1,X     ;TEAM SCORE.
        PHA             ;HOLD IT.
        AND #$0F        ;PUT '5' IN
        ORA #$50        ;HI-NIBBLE.
        STA TEAM2+1,Y   ;STORE DIGIT.
        PLA             ;RESTORE IT.
        LSR A           ;SHIFT HI-
        LSR A           ;NIBBLE TO LO-
        LSR A           ;NIBBLE.
        LSR A           ;PUT '5' IN
        ORA #$50        ;HI-NIBBLE.
        STA TEAM2,Y     ;STORE IT.
        DEX             ;IF NOT DONE,
        BPL CALC8       ;CONTINUE.

        CLD             ;CLEAR DECIMAL
        RTS             ;ALL DONE...

*       TURN OFF SOUND

SHUTUP  LDA #$00    ;DEACTIVATE
        STA AUDC3   ;SOUND REGISTERS
        STA AUDC4   ;3 & 4.
        STA LSOUND  ;DEACTIVATE
        STA XSOUND  ;EXPLOSIONS,
        STA CSOUND  ;CANNON & LASERS.
        RTS         ;ALL DONE...

*       DLI ROUTINES

*       GAME BOARD DLIs

DLI     CLD             ;CLEAR DEC.
        PHA             ;SAVE ACC.
        LDA GEVX+0      ;GET X-COORDS
        STA HPOSP0      ;OF ALL ACTIVE
        LDA GEVX+1      ;VESSELS &
        STA HPOSP1      ;STORE INTO PM
        LDA GEVX+2      ;HORIZONTAL
        STA HPOSP2      ;REGISTERS.
        LDA GEVX+3      ;
        STA HPOSP3      ;
        LDA GEVX+4      ;
        STA HPOSM3      ;
        CLC             ;
        ADC #$02        ;
        STA HPOSM2      ;
        CLC             ;
        ADC #$02        ;
        STA HPOSM1      ;
        CLC             ;
        ADC #$02        ;
        STA HPOSM0      ;
        LDA #LOW DLI2   ;SET UP FOR
        STA VDSLST      ;NEXT DISPLAY
        LDA #HIGH DLI2  ;LIST INTER-
        STA VDSLST+1    ;RUPT REQUEST.
        PLA             ;RESTORE ACC.
        RTI             ;ALL DONE...

DLI2    PHA             ;SAVE ACC.
        LDA #$00        ;MAKE X-COORDS
        STA HPOSP0      ;OF ALL ACTIVE
        STA HPOSP1      ;VESSELS EQUAL
        STA HPOSP2      ;TO ZERO. THIS
        STA HPOSP3      ;PROVIDES A
        STA HPOSM0      ;NICE BORDER
        STA HPOSM1      ;AT THE
        STA HPOSM2      ;BOTTOM.
        STA HPOSM3      ;
        PLA             ;RESTORE ACC.
        RTI             ;ALL DONE...

* TITLE SCREEN DLIs

DLIDLI  PHA             ;SAVE ACC.
        TXA             ;SAVE THE
        PHA             ;X-REG.
        LDX #$07        ;LOAD X-REG.
        LDA DSHIFT      ;OKAY, NOW
        STA CSHIFT      ;I'LL LET
DLIXX   LDA CSHIFT      ;YOU GUYS
        STA WSYNC       ;FIGURE OUT
        STA COLPF0+1    ;HOW I GOT
        LDA CSHIFT      ;THAT SUPER
        CLC             ;DUPER STU-
        ADC #$02        ;PENDOUS
        STA CSHIFT      ;FANCY
        DEX             ;COLOR CHANGE
        BPL DLIXX       ;(SO THERE!)
        LDA #LOW DLIXY  ;SET UP FOR
        STA VDSLST+0    ;NEXT DLI
        LDA #HIGH DLIXY ;(IF YOU
        STA VDSLST+1    ;DON'T MIND).
        DEC BLAH        ;DEC TIMER.
        BPL DLIZZ       ;BRANCH!
        LDA STRUCT      ;RESET
        STA BLAH        ;TIMERS.
        INC BLAH        ;INCREMENT
        ASL BLAH        ;& SHIFT
        INC DSHIFT      ;REGISTERS.
DLIZZ   PLA             ;RESTORE
        TAX             ;X-REG.
        PLA             ;RESTORE ACC.
        RTI             ;LATER Y'ALL!

DLIXY   PHA             ;SAVE ACC.
        LDA COLOR0+1    ;RESTORE OLD
        STA COLPF0+1    ;COLOR REG.
        PLA             ;RESTORE ACC.
        RTI             ;LATER Y'ALL!

:XHOLD  DB      0       ;X-REG. TEMP.
CSHIFT  DB      0       ;COLOR SHIFT.
DSHIFT  DB      0       ;COLOR SHIFT.
BLAH    DB      0       ;DUM TIMER.

:LSTDB  DW      DLIST   ;LO/HI DISPLAY
        DW      DLIST2  ;LIST BYTES.

:DLIDB  DW      DLI     ;LO/HI DLI
        DW      DLIDLI  ;BYTES.

*       PLAYER SCORE PLACEMENT BYTES

PUTSCR  DB 1,13,3,15,5,17

*       TEAM SCORE PLACEMENT BYTES

PUTTME  DB 7,9,11

*       SCORE PLACEMENT MASKS

SCMASK  DB $0E,$0F


 TITLE  'OUT OF BOUNDS CHECK'

BOUNDS  PROC

        LDX #$04        ;CHECK EVERYBODY

*       CHECK Y-COORDS

BOUND3  CPX #$02        ;A BACTERION?
        BCS CHECKX      ;IF SO LEAVE IT.
        LDA GEVY,X      ;IS PLAYER OUT
        CMP LOWY        ;OF BOUNDS?
        BCS BOUND5      ;NO, CONTINUE.
BOUND4  LDY GEVDIR,X    ;OUT OF BOUNDS!
        LDA REFLEY,Y    ;GET REFLEX
        STA GEVDIR,X    ;ANGLE & STORE.
        JSR MOVSUB      ;MOVE ONCE MORE
        JMP CHECKX      ;& CHECK X-COORD
BOUND5  CMP HIGHY       ;BOUNDARY.
        BCS BOUND4      ;

*       CHECK X-COORDS

CHECKX  LDA GEVX,X      ;OUT OF BOUNDS
        CMP LOWX        ;ON X-COORDS?
        BCS BOUND7      ;NO, CONTINUE.
BOUND6  LDY GEVDIR,X    ;OUT OF BOUNDS!
        LDA REFLEX,Y    ;GET REFLEX
        STA GEVDIR,X    ;ANGLE & STORE.
        JSR MOVSUB      ;MOVE ONCE MORE
        JMP CHECKY      ;& CHECK NEXT.
BOUND7  CMP HIGHX       ;
        BCS BOUND6      ;
CHECKY  DEX             ;CHECK NEXT
        BPL BOUND3      ;VESSEL.
        RTS             ;ALL DONE.

LOWX    DB      44      ;LOWEST X-COORD
LOWY    DB      32      ;LOWEST Y-COORD
HIGHX   DB      208     ;HIGHEST X-COORD
HIGHY   DB      215     ;HIGHEST Y-COORD

*  REFLEX DIRECTIONS FOR Y-COORDS

REFLEY  DB 08,07,06,05,04,03,02,01
        DB 00,15,14,13,12,11,10,09

*  REFLEX DIRECTIONS FOR X-COORDS

REFLEX  DB 00,15,14,13,12,11,10,09
        DB 08,07,06,05,04,03,02,01


 TITLE 'TITLE SCREEN DISPLAY LIST'

DLIST2  DB $70,$70,$70,$70,$46
        DW PLAYR1
        DB $46
        DW PLAYR2
        DB $70,$70,$70,$F0,$46
        DW TITLE
        DB $B0,$46
        DW TITLE2
        DB $50,$46
        DW NAME1
        DB $20,$46
        DW NAME2
        DB $20,$46
        DW NAME3
        DB $20,$46
        DW NAME4
        DB $70,$70,$46
        DW NAME5
        DB $70,$70,$46
        DW TEAM1
        DB $46
        DW TEAM2
        DB $41
        DW DLIST2


 TITLE 'BACTERION! STRATEGY ROUTINE'

STRAT   PROC

        LDX GEVNUM      ;BACTERION #

*  SHOULD BACTERION! ATTACK PLAYER?

        LDA STOP,X      ;IS BACTERION
        BEQ :CONT       ;IN A COMA?
        RTS             ;YES! QUIT!
:CONT   LDA GEVCEL,X    ;CELL IN TOW?
        BMI :CONT2      ;YES! QUIT!
        LDY NOPLAY      ;# OF PLAYERS.
:CONT0  LDA STOP,Y      ;THIS PLR ICED?
        BNE :CONT1      ;YES! CONTINUE!
        LDA GEVX,X      ;IS X-COORD OF
        SEC             ;PLAYER WITHIN
        SBC GEVX,Y      ;ATTACKING
        JSR ABS         ;RANGE?
        CMP FRANGE      ;
        BCS :CONT1      ;NO! QUIT!
        LDA GEVY,X      ;IS Y-COORD OF
        SEC             ;PLAYER WITHIN
        SBC GEVY,Y      ;ATTACKING
        JSR ABS         ;RANGE?
        CMP FRANGE      ;
        BCS :CONT1      ;NO! QUIT!
        LDA GEVX,Y      ;PLR X-COORD IS
        STA :TARX       ;TARGET X-COORD
        LDA GEVY,Y      ;PLR Y-COORD IS
        STA :TARY       ;TARGET Y-COORD
        JSR LASERS      ;SCRAP PLAYER!
        LDA #$01        ;KABOOM!!!
        BNE STRAT2      ;
:CONT1  DEY             ;CHECK NEXT
        BPL :CONT0      ;PLAYER. IF ANY.
:CONT2  LDA TARX,X      ;SAVE TARGET
        STA :TARX       ;X-COORD.
        LDA TARY,X      ;SAVE TARGET
        STA :TARY       ;Y-COORD.
        LDA #$00        ;
STRAT2  STA ATTACK,X    ;ATTACK STATUS

*       FIND THE MARK

*       AXIS TESTS

STRAT3  LDA :TARX   ;IS THE TARGET
        CMP GEVX,X  ;(RELATIVE TO
        BEQ AXIS5   ;ATTACKING
        BCS AXIS7   ;BACTERION)

        LDA :TARY   ;ON THE X
        CMP GEVY,X  ;OR Y AXIS? THAT
        BNE AXIS4   ;IS THE QUESTION!

        LDA #$04    ;WHO EVER HEARD
        BNE QUIT    ;OF CENSORING

AXIS4   BCS QUAD3   ;ASSEMBLY LAN-
        BCC QUAD2   ;GUAGE LISTINGS?

AXIS5   LDA :TARY   ;BACTERION! IS
        CMP GEVY,X  ;DEDICATED TO MY
        BEQ EXIT    ;FRIENDS IN GOOD
        BCS AXIS6   ;OLD TEANECK...

        LDA #$00    ;IN CASE YOU
        BEQ QUIT    ;PEOPLE DON'T

AXIS6   LDA #$08    ;KNOW WHO YOU ARE,
        BNE QUIT    ;HERE'S A LIST

AXIS7   LDA :TARY   ;OF CODE-NAMES...
        CMP GEVY,X  ;SIR HEX
        BNE AXIS8   ;THE QUAB RUNNER

        LDA #12     ;THE SILICON
        BNE QUIT    ;PIRATE

AXIS8   BCS QUAD4   ;TAI-FIGHTER &
        BCC QUAD1   ;HOME BOY ROGER D.

*       FIRST QUADRANT

QUAD1   JSR DELTAS  ;
        BEQ QUAD12  ;
        BCC QUAD13  ;WE KNOW THE TAR-
        LDA #13     ;GET IS IN THE
        BNE QUIT    ;1st QUAD. DECIDE
QUAD12  LDA #14     ;ON A DIRECTION.
        BNE QUIT    ;(13,14,15)
QUAD13  LDA #15     ;
        BNE QUIT    ;BRANCH!

*       SECOND QUADRANT

QUAD2   JSR DELTAS  ;
        BEQ QUAD22  ;
        BCC QUAD23  ;WE KNOW THE TAR-
        LDA #03     ;GET IS IN THE
        BNE QUIT    ;2nd QUAD. DECIDE
QUAD22  LDA #02     ;ON A DIRECTION.
        BNE QUIT    ;(1,2,3)
QUAD23  LDA #01     ;
        BNE QUIT    ;BRANCH!

*       THIRD QUADRANT

QUAD3   JSR DELTAS  ;
        BEQ QUAD32  ;
        BCC QUAD33  ;WE KNOW THE TAR-
        LDA #05     ;GET IS IN THE
        BNE QUIT    ;3rd QUAD. DECIDE
QUAD32  LDA #06     ;ON A DIRECTION.
        BNE QUIT    ;(5,6,7)
QUAD33  LDA #07     ;
        BNE QUIT    ;BRANCH!

*       FOURTH QUADRANT

QUAD4   JSR DELTAS  ;
        BEQ QUAD42  ;
        BCC QUAD43  ;WE KNOW THE TAR-
        LDA #11     ;GET IS IN THE
        BNE QUIT    ;4th QUAD. DECIDE
QUAD42  LDA #10     ;ON A DIRECTION.
        BNE QUIT    ;(9,10,11)
QUAD43  LDA #09     ;
        BNE QUIT    ;BRANCH!

EXIT    LDA #$FF    ;TARGET REACHED

QUIT    STA GEVDES,X ;SAVE DESIRED
        RTS          ;DIRECTION. BYE!

*       CALCULATE COORDINATE DELTAS

DELTAS  LDA :TARY   ;TARGET Y-COORD.
        SEC         ;SUBTRACT FROM
        SBC GEVY,X  ;BACTERION Y-CORD.
        JSR ABS     ;ABSOLUTE VALUE.
        STA :DELTA  ;SAVE IT.
        LDA :TARX   ;TARGET X-COORD.
        SEC         ;SUBTRACT FROM
        SBC GEVX,X  ;BACTERION X-CORD.
        JSR ABS     ;ABSOLUTE VALUE.
        CMP :DELTA  ;COMPARE TO Y
        RTS         ;DELTA & RETURN.

*       TURN ENEMY BACTERION!s

TURN    DEC TURNT       ;TIME TO TURN?
        BPL TRTS        ;NO! LATER!
        LDX TYPES       ;RESTORE TURN
        LDA TURNDB-1,X  ;TIME FROM
        STA TURNT       ;TURN DATABASE.
        LDX #$04        ;HANDLE ALL.
TURN0   LDA STOP,X      ;VESSEL ICED?
        BNE PL4         ;YES! LATER!

        LDA #$03    ;GET RANDOM AMOUNT
        JSR RANDO   ;FOR DIRECTIONAL
        TAY         ;TURN.
```

```
        INY         ;
        LDA GEVDES,X ;DESIRED DIRECTION.
        CMP GEVDIR,X ;ACTUAL DIRECTION.
        BEQ PL4      ;IF = QUIT!
        BCS TURN05   ;ACT. > DES.
        JSR SUBDIR   ;SUB. THE TWO.
        CMP #$08     ;IS DIFF. > 8?
        BCS TURN06   ;YES! BRANCH!
TURN04  TYA          ;INVERT AMOUNT
        EOR #$FF     ;OF TURN (MAKE
        TAY          ;IT NEGATIVE)
        INY          ;
        BMI TURN06   ;BRANCH!
TURN05  JSR SUBDIR   ;SUB. THE TWO.
        CMP #$08     ;IF DIFF. > 8?
        BEQ TURN06   ;DIFF = 8!
        BCS TURN04   ;DIFF > 8!
TURN06  TYA          ;ADD AMOUNT OF
        CLC          ;TURN TO BAC-
        ADC GEVDIR,X ;TERION'S DIR-
        JSR WRAP     ;ECTION & SAVE.
        STA GEVDIR,X ;

PL4     DEX          ;TURN NEXT
        CPX #$01     ;BACTERION!
        BNE TURN0    ;IF ANY.
TRTS    RTS          ;BOOGIE OUTTA HERE!

SUBDIR  LDA GEVDES,X ;DESIRED
        SEC          ;SUBTRACTED FROM
        SBC GEVDIR,X ;ACTUAL
        JMP ABS      ;ABSOLUTE VALUE.

*       MOVE ENEMY BACTERION!s

MOVE    LDX #$04          ;HANDLE ALL.
MOVE0   DEC MOVET,X       ;TIME TO MOVE?
        BPL MOVENX        ;NO! LATER!
        LDY TYPES         ;RESTORE MOVE
        LDA MOVEDB-1,Y    ;TIMER.
        LDY GEVCEL,X      ;CELL IN TOW?
        BPL MOVE1         ;NO! MOVEFAST!
        ASL A             ;YES! MOVE A
        ASL A             ;TIMES AS SLOW!
MOVE1   STA MOVET,X       ;SAVE MOVE
        JSR MOVSUB        ;TIMER & MOVE!
MOVENX  DEX               ;HANDLE NEXT
        CPX #$01          ;BACTERION!
        BNE MOVE0         ;IF ANY.
        RTS               ;OUTTA HERE!

*       TRACK HEISTED CELLS

TRACK   LDX #$04      ;HANDLE ALL.
TRK0    LDA STOP,X    ;VESSEL ON ICE?
        BNE TRK1      ;YES! BYPASS!

        LDA GEVCEL,X ;CELL IN TOW?
        BPL TRKTRK   ;NO! QUIT!
        LDA GEVY,X   ;IS VESSEL ALL
        CMP #15      ;THE WAY OFF-
        BCC OFFSCR   ;SCREEN?
        CMP #235     ;
        BCC TRKTRK   ;NO! QUIT!

* SUCCESSFUL HEISTING OF A CELL

OFFSCR  INC GEVESC    ;INC. ESCAPEES
        DEC TOTCEL    ;DEC. # OF CELLS
        LDA #$01      ;HALT THIS
        STA STOP,X    ;BACTERION.
        BNE TRK1      ;BRANCH!

TRKTRK  LDA ATTACK,X ;ATTACKING?
        BNE TRK1     ;YES! SKIP!
        LDA GEVDES,X ;AT TARGET?
        BPL TRK1     ;YES! SKIP!

        LDA ESCAPE,X ;ATTACKING PLR?
        BEQ TRK1     ;YES! BRANCH!
        STA TARY,X   ;NO! SET Y-TARGET

        JSR PICKX    ;PICK RANDOM X-TARGET.
        STA TARX,X   ;SAVE IT.
        JSR PICKDR   ;GET A DIRECTION
        LDA GEVCEL,X ;GIVE CELL A
        ORA #$80     ;HEISTED STATUS.
        STA GEVCEL,X ;

TRK1    DEX          ;HANDLE NEXT
        CPX #$01     ;BACTERION!
        BNE TRK0     ;IF ANY.
        RTS          ;GET LOST LOSER!

* ACCUMULATOR ABSOLUTE VALUE FUNCTION

ABS     BPL RABS
        EOR #$FF
        CLC
        ADC #$01
RABS    RTS

*       DIRECTIONAL WRAP AROUND

WRAP    BPL PWRAP ;DIRECTION > 0?
        CLC       ;NO! ADD 16 TO IT.
        ADC #16   ;
        RTS       ;GET LOST LOSER!
PWRAP   CMP #16   ;DIRECTION < 16?
        BCC WRTS  ;NO! SUBTRACT 16
        SEC       ;FROM IT.
        SBC #16   ;
WRTS    RTS       ;GET LOST LOSER!

*       GENERAL MOVEMENT ANALYSIS

MOVSUB  LDA STOP,X    ;COMATOSE?
        BNE MOV1      ;YES! QUIT!
        LDA GEVDES,X ;TARGET REACHED?
        BMI MOV1      ;YES! QUIT!
        LDY GEVDIR,X ;GET DIRECTION.
        LDA DELX,Y   ;ADD X-COORD ADD
        STA :ADDX    ;ON FOR GIVEN
        CLC          ;DIRECTION.
        ADC GEVX,X   ;
        STA GEVX,X   ;SAVE IT.
        LDA DELY,Y   ;ADD Y-COORD ADD
        STA :ADDY    ;ON FOR GIVEN
        CLC          ;DIRECTION.
        ADC GEVY,X   ;
        STA GEVY,X   ;SAVE IT.
        LDA GEVCEL,X ;CELL IN TOW?
        BPL MOV1     ;NO! QUIT!
        AND #$7F     ;YES!
        TAY          ;
        LDA :ADDX    ;ADD X-COORD ADD
        CLC          ;ON FOR GIVEN
        ADC CELLNX,Y ;DIRECTION TO
        STA CELLNX,Y ;CELL'S X-COORD.
        LDA :ADDY    ;ADD Y-COORD ADD
        CLC          ;ON FOR GIVEN
        ADC CELLNY,Y ;DIRECTION TO
        STA CELLNY,Y ;CELL'S Y-COORD.
        LDA #$01     ;CELL MOVEMENT
        STA CELLMV,Y ;STATUS SET.
MOV1    RTS          ;SEE YA LATER...

* VAPORIZE PLAYERS WITH DEADLY LASERS

LASERS  DEC GEVFRE,X ;FIRE TIMER = 0?
        BPL XITLSR   ;NO! QUIT!
        LDA FIRETM   ;RESTORE FIRE
        STA GEVFRE,X ;TIMER.
        LDA LSRDIR,X ;ALREADY FIRING?
        BNE XITLSR   ;YES! QUIT!
        LDA GEVDES,X ;SAVE DESIRED
        PHA          ;DIRECTION.
        JSR STRAT3   ;GET LASER DIRECTON.
        TAY          ;PUT IN Y-REG.
        PLA          ;RESTORE DESIRED
        STA GEVDES,X ;DIRECTION.
        CPY #$FF     ;LASER AT TARGET?
        BEQ XITLSR   ;YES! QUIT!
        LDA LSRBSE,Y ;FINE TUNE LASER
        STA LSRDIR,X ;DIRECTION.
        LDA #$0A     ;LASER BOLT
        STA LSRTME,X ;LIFETIME.
        LDA #14      ;SET UP LASER
        STA LSOUND   ;SOUND.
XITLSR  RTS          ;JUMP IN A LAKE!

*       LASER DIRECTIONAL FINE TUNING

LSRBSE  DB 1,8,8,8,4,6,6,6,2
        DB 7,7,7,3,3,3,3,3

:TARX   DB      0 ;X-COORD TARGET
:TARY   DB      0 ;Y-COORD TARGET
:ADDX   DB      0 ;X-COORD ADD ON
:ADDY   DB      0 ;Y-COORD ADD ON
:DELTA  DB      0 ;TAR. Y - ACT. Y

*       X-COORD DIRECTIONAL ADD ONs

DELX    DB +0,-1,-2,-2,-2,-2,-2,-1
        DB +0,+1,+2,+2,+2,+2,+2,+1

*       Y-COORD DIRECTIONAL ADD ONs

DELY    DB -2,-2,-2,-1,+0,+1,+2,+2
        DB +2,+2,+2,+1,+0,-1,-2,-2

*       TURN DATABASE

TURNDB  DB 0,0,0,0,0,0,0,0

*       MOVEMENT DATABASE

MOVEDB  DB 0,0,0,0,0,0,0,0
```

```
 TITLE 'BACTERION! PM DRAW ROUTINE'

DRAW    PROC

        LDA #$04        ;HANDLE ALL.
        STA COUNT       ;VESSELS (0-4)
DRAW1   LDX COUNT       ;VESSEL #.
        LDY TYPE,X      ;TYPE (0-7).
        LDA PHASE,X     ;SHAPE PHASE.
        CMP INDEX+1,Y   ;PHASE EXCEE-
        BCC DRAW3       ;DED?
DRAW2   LDA INDEX,Y     ;YES! CORRECT
        STA PHASE,X     ;PHASE.
DRAW3   CMP INDEX,Y     ;PHASE EXCEE-
        BCC DRAW2       ;DED?
        ASL A           ;NO! GET OFF-
        TAY             ;SET TO DATA
        LDA OFFSET,Y    ;FOR THIS
        STA DRWLO       ;PHASE.
        LDA OFFSET+1,Y  ;
        STA DRWLO+1     ;
        LDA COUNT       ;FIND WHERE TO
        ASL A           ;PLACE PHASE
        TAY             ;DATA. (DEP-
        LDA PLBSE,Y     ;ENDS ON WHICH
        STA PLLO        ;VESSEL # WE
        LDA PLBSE+1,Y   ;ARE DRAWING).
        STA PLLO+1      ;
        LDA #$07        ;PREPARE TO
        STA PTR1        ;READ 8 BYTES.
        CLC             ;FIND OUT
        ADC GEVY,X      ;WHERE TO PUT
        STA PTR2        ;DATA.
DRAW4   LDY PTR1        ;GET PHASE
        LDA (DRWLO),Y   ;DATA.
        LDY PTR2        ;PUT INTO PM
        STA (PLLO),Y    ;LOCATIONS.
        DEC PTR2        ;RESET
        DEC PTR1        ;POINTERS.
        BPL DRAW4       ;ALL DONE?
        LDA #$07        ;YES! SET UP
        STA COUNT2      ;FOR 8 BYTES.
DRAW5   LDX COUNT       ;BEGIN ERASING
        LDA GEVY,X      ;DATA AT TOP
        LDX COUNT2      ;& BOTTOM OF
        CLC             ;VESSEL. THIS
        ADC ERADD,X     ;ACCOUNTS FOR
        TAY             ;WHEN A VESSEL
        LDA #$00        ;MOVES VERTI-
        STA (PLLO),Y    ;CALLY AS WE
        DEC COUNT2      ;MUST PRE-
        BPL DRAW5       ;VENT DATA
        DEC COUNT       ;OVERLAP.
        BPL DRAW1       ;YEAH SO WHAT!

DRAW6   DEC PHTIME ;DEC PHASE TIMER.
        BPL DRAW7  ;QUIT IF NOT 0.
        LDA #$04   ;RESET PHASE TIMER
        STA PHTIME ;& STORE.

        TAX             ;HANDLE ALL (0-4)
PHS4    CPX #$02        ;A BACTERION?
        BCS PHS5        ;YES! SKIP IT!
        LDA TYPE,X      ;IS IT EXPLODING?
        CMP #$07        ;
        BNE PHS7        ;NO! SKIP IT!
PHS5    LDA PHASE,X     ;
        CMP #47         ;(INDEX+8)-1
        BEQ PHS7        ;
        INC PHASE,X     ;INC VESSEL PHASE
        CMP #46         ;(INDEX+8)-2
        BNE PHS7        ;
        CPX #$02        ;IS VESSEL A
        BCC PHS7        ;PLAYER?
        INC GEVESC      ;NO! INC # OF ESCAPES
PHS7    DEX             ;MOVE ON TO
        BPL PHS4        ;NEXT...

DRAW7   RTS ;TIME TO BOOGIE...

COUNT   DB 0 ;BYTE DRAW COUNTER
COUNT2  DB 0 ;BYTE DRAW COUNTER
PTR1    DB 0 ;POINTER TO PHASE DATA
PTR2    DB 0 ;POINTER TO PM AREA
PHTIME  DB 3 ;VESSEL PHASE TIMER

*       TOP/BOTTOM ERASURE OFFSETS

ERADD   DB -4,-3,-2,-1,+8,+9,+10,+11

*       INDEX TO VESSEL PHASE DATA

INDEX   DB 0,16,20,24,28,31,35,39,48

*       LO/HI BYTES TO PM AREA

PLBSE   DW PLAY0
        DW PLAY1
        DW PLAY2
        DW PLAY3
        DW MISS

*       OFFSET TO VESSEL PHASE DATA

OFFSET  DW :ROT00       ;0
        DW :ROT01
        DW :ROT02
        DW :ROT03
        DW :ROT04
        DW :ROT05
        DW :ROT06
        DW :ROT07
        DW :ROT08
        DW :ROT09
        DW :ROT10
        DW :ROT11
        DW :ROT12
        DW :ROT13
        DW :ROT14
        DW :ROT15

        DW :GEV10       ;16
        DW :GEV11
        DW :GEV12
        DW :GEV11

        DW :GEV20       ;20
        DW :GEV21
        DW :GEV22
        DW :GEV21

        DW :GEV30       ;24
        DW :GEV31
        DW :GEV32
        DW :GEV31

        DW :GEV40       ;28
        DW :GEV41
        DW :GEV42

        DW :GEV50       ;31
        DW :GEV51
        DW :GEV52
        DW :GEV51

        DW :GEV60       ;35
        DW :GEV61
        DW :GEV62
        DW :GEV63

        DW :EXP0        ;39
        DW :EXP1
        DW :EXP2
        DW :EXP3
        DW :EXP4
        DW :EXP5
        DW :EXP6
        DW :EXP7
        DW :EXP8

                        ;48
*       TANK ROTATION 0

:ROT00  DB $10,$10,$10,$38
        DB $54,$82,$44,$44

*       TANK ROTATION 1

:ROT01  DB $20,$20,$10,$1E
        DB $19,$21,$20,$10

*       TANK ROTATION 2

:ROT02  DB $80,$40,$20,$1F
        DB $19,$10,$10,$18

*       TANK ROTATION 3

:ROT03  DB $00,$00,$C6,$39
        DB $18,$10,$10,$0C

*       TANK ROTATION 4

:ROT04  DB $04,$0B,$10,$F0
        DB $10,$0B,$04,$00

*       TANK ROTATION 5

:ROT05  DB $0C,$10,$10,$18
        DB $39,$C6,$00,$00

*       TANK ROTATION 6

:ROT06  DB $18,$10,$10,$19
        DB $1F,$20,$40,$80

*       TANK ROTATION 7

:ROT07  DB $10,$20,$21,$19
        DB $1E,$10,$20,$20

*       TANK ROTATION 8

:ROT08  DB $44,$44,$82,$44
        DB $38,$10,$10,$10

*       TANK ROTATION 9

:ROT09  DB $08,$04,$84,$98
        DB $78,$08,$04,$04

*       TANK ROTATION 10
```

```
:ROT10  DB $18,$08,$08,$98
        DB $F8,$04,$02,$01

*       TANK ROTATION 11

:ROT11  DB $30,$08,$08,$18
        DB $9C,$63,$00,$00

*       TANK ROTATION 12

:ROT12  DB $20,$D0,$08,$0F
        DB $08,$D0,$20,$00

*       TANK ROTATION 13

:ROT13  DB $00,$00,$63,$1C
        DB $18,$08,$08,$30

*       TANK ROTATION 14

:ROT14  DB $01,$02,$04,$F8
        DB $98,$08,$08,$18

*       TANK ROTATION 15

:ROT15  DB $04,$04,$08,$78
        DB $98,$84,$04,$08

*       BACTERION! #1

:BEV10  DB $18,$00,$24,$81
        DB $81,$24,$00,$18

:BEV11  DB $18,$42,$24,$81
        DB $81,$24,$42,$18

:BEV12  DB $99,$42,$24,$81
        DB $81,$24,$42,$99

*       BACTERION! #2

:BEV20  DB $00,$00,$3C,$24
        DB $24,$3C,$00,$00

:BEV21  DB $00,$66,$42,$18
        DB $18,$42,$66,$00

:BEV22  DB $C3,$81,$00,$18
        DB $18,$00,$81,$C3

*       BACTERION! #3

:BEV30  DB $3C,$42,$A5,$81
        DB $81,$A5,$42,$3C

:BEV31  DB $00,$18,$24,$42
        DB $42,$24,$18,$00

:BEV32  DB $00,$00,$18,$3C
        DB $3C,$18,$00,$00

*       BACTERION! #4

:BEV40  DB $00,$00,$18,$24
        DB $24,$18,$00,$00

:BEV41  DB $00,$18,$00,$5A
        DB $5A,$00,$18,$00

:BEV42  DB $18,$00,$18,$A5
        DB $A5,$18,$00,$18

*       BACTERION! #5

:BEV50  DB $20,$20,$E4,$18
        DB $18,$27,$04,$04

:BEV51  DB $00,$24,$66,$18
        DB $18,$66,$24,$00

:BEV52  DB $04,$04,$27,$18
        DB $18,$E4,$20,$20

*       BACTERION! #6

:BEV60  DB $0C,$40,$90,$25
        DB $25,$88,$40,$0C

:BEV61  DB $18,$00,$99,$A1
        DB $04,$18,$42,$24

:BEV62  DB $30,$02,$19,$84
        DB $A0,$19,$02,$30

:BEV63  DB $24,$42,$08,$24
        DB $A5,$91,$00,$18

*       DETONATION

:EXP0   DB $00,$00,$00,$18
        DB $18,$00,$00,$00

:EXP1   DB $00,$00,$08,$38
        DB $1C,$10,$00,$00

:EXP2   DB $00,$08,$08,$78
        DB $1E,$10,$10,$00

:EXP3   DB $08,$08,$2C,$E0
        DB $07,$34,$10,$10

:EXP4   DB $08,$4A,$24,$C0
        DB $03,$24,$52,$10

:EXP5   DB $89,$42,$24,$80
        DB $91,$24,$42,$91

:EXP6   DB $81,$42,$00,$00
        DB $00,$00,$42,$81

:EXP7   DB $81,$00,$00,$00
        DB $00,$00,$00,$81

:EXP8   DB $00,$00,$00,$00
        DB $00,$00,$00,$00

*TYPE   - TYPE OF TANK BEING DRAWN
*         0 - PLAYER$ # 1 & 2
*       1-6 - GEV$ 1,2,3,4,5,6
*         7 - DETONATION SEQUENCE
*PHASE  - PHASE # OF BACTERION!$
*     0-15 - PLAYER$ # 1 & 2
*    16-19 - BACTERION! # 1
*    20-23 - BACTERION! # 2
*    24-27 - BACTERION! # 3
*    28-30 - BACTERION! # 4
*    31-34 - BACTERION! # 5
*    35-38 - BACTERION! # 6
*    39-48 - EXPLOSION SEQUENCE
```

```
 TITLE 'BACTERION TITLE SCREEN'

        PROC

*       BACTERION!

TITLE   DB 0,0,0,0,0
        DB 098,097,099,116,101
        DB 114,105,111,110,065
        DB 0,0,0,0,0

*       THE PLAGUE OF 2369

TITLE2  DB 000,180,168,165,00,176,172
        DB 161,167,181,165,00,175,166
        DB 00,210,211,214,217,00

*       TEAM SCORE

TEAM1   DB 0,0,0,0,0
        DB 180,165,161,173,000
        DB 179,163,175,178,165
        DB 0,0,0,0,0

TEAM2   DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0

PLAYR1  DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0

PLAYR2  DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0
        DB 0,0,0,0,0

*       BY

NAME1   DB 00,00,00,00,00
        DB 00,00,00,00,34
        DB 57,00,00,00,00
        DB 00,00,00,00,00

*       KYLE PEACOCK

NAME2   DB 0,0,0,0
        DB 235,249,236,229,000,240
        DB 229,225,227,239,227,235
        DB 0,0,0,0

*       WITH

NAME3   DB 0,0,0,0,0,0,0,0,0
        DB 55,41,52,40
        DB 0,0,0,0,0,0,0,0,0

*       TOM HUDSON

NAME4   DB 0,0,0,0,0
        DB 244,239,237,000,232
        DB 245,228,243,239,238
        DB 0,0,0,0,0

*       ANALOG COMPUTING

NAME5   DB $00,$00,$A1,$AE,$A1
        DB $AC,$AF,$A7,$00,$A3
        DB $AF,$AD,$B0,$B5,$B4
        DB $A9,$AE,$A7,$00,$00
```

```
 TITLE 'JOYSTICK READING'

STICKS  PROC

        LDX NOPLAY      ;# OF PLAYERS

ACT4    LDA STOP,X      ;IS PLAYER DEAD?
        BEQ ACT5        ;NO, CONTINUE.
        DEC STOP,X      ;DEC. DEATH TIME
        BNE NXTACT      ;& CONTINUE.
        JSR SETPLR      ;IF DEATH TIME=0
        JMP NXTACT      ;REINCARNATE.

ACT5    JSR MOVPLR      ;MOVE PLAYER

        DEC TURNIT,X    ;DEC. TURN TIME
        BPL NXTACT      ;IF <> 0 CONT.
        LDA #$03        ;RESET TURN
        STA TURNIT,X    ;TIME & STORE.

        LDY STICK0,X    ;READ JOYSTICK.
        LDA GEVDIR,X    ;UPDATE DIRECT-
        CLC             ;ION ACCORDING
        ADC DRHASH,Y    ;TO JOYSTICK.
        JSR WRAP        ;TEST FOR WRAP
        STA GEVDIR,X    ;AROUND &
        STA PHASE,X     ;SAVE.

NXTACT  DEX             ;HANDLE NEXT
        BPL ACT4        ;PLAYER.
        RTS             ;ALL DONE...

*       GENERAL MOVEMENT ROUTINE

MOVPLR  DEC SPEED,X     ;TIME TO MOVE?
        BPL RTSMOV      ;NO, CONTINUE.
        LDA STICK0,X    ;YES-READ STICK.
        CMP #14         ;FORWARD MOTION.
        BEQ :MOV4       ;
        CMP #10         ;FORWARD MOTION.
        BEQ :MOV4       ;
        CMP #06         ;FORWARD MOTION.
        BNE :MOV5       ;
:MOV4   DEC CSPEED,X    ;DEC. MOTION
        JMP :MOV6       ;TIMER.
:MOV5   INC CSPEED,X    ;INC. MOTION
:MOV6   LDY CSPEED,X    ;TIMER.
        LDA REHASH,Y    ;DETECT OVERFLOW
        STA CSPEED,X    ;& CORRECT (IF
        STA SPEED,X     ;ANY.) THEN SAVE
        LDA SPEED,X     ;MOTION TIMER.
        CMP DRHASH-1    ;
        BEQ RTSMOV      ;
        JSR MOVSUB      ;MOVE TO CORRECT
RTSMOV  RTS             ;ALL DONE...

TURNIT  DB 0,0  ;PLAYERS' TURN TIMER.

*       SPEED LIMITATION DATABASE

REHASH  DB 2,2,2,3,4,5,6,7,8,8,8

*       JOYSTICK/DIRECTION ADD ONS

DRHASH  DB 0,0,0,0,0,-1,-1,-1
        DB 0,+1,+1,+1,0,0,0,0
```

```
 TITLE 'COLLISION DETECTION'

COLLIDE PROC

*       SHIP COLLISION

        LDX NOPLAY      ;# OF PLAYERS
COL5    LDA STOP,X      ;IS THIS PLR ICED?
        BNE COLXX       ;YES! MOVE ALONG.
        LDA #$01        ;NO! CHECK FOR
        STA IDIE        ;COLLISION...
        LDA P0PL,X      ;
        AND #$02        ;PLR/PLR
        BEQ COL7        ;NO COLLISION.
        JSR KILLME      ;SMASH! BANG!
COL7    INC IDIE        ;CHECK FOR ANOTHER
        LDA P0PL,X      ;COLLISION...
        AND #$04        ;PLR/BACTERION 1
        BEQ COL8        ;NO COLLISION.
        JSR KILLME      ;OUCH! CRASH!
COL8    INC IDIE        ;CHECK FOR ANOTHER
        LDA P0PL,X      ;COLLISION...
        AND #$08        ;PLR/BACTERION 2
        BEQ COL9        ;NO COLLISION.
        JSR KILLME      ;DING! DONG!
COL9    INC IDIE        ;CHECK FOR ANOTHER
        LDY #$03        ;COLLISION
COL10   TXA             ;W/MISSILES...
        CLC             ;
        ADC #$01        ;
        AND M0PL,Y      ;PLR/BACTERION 3
        BEQ COL11       ;NO COLLISION...
        JSR KILLME      ;BING! ZAP!
COL11   DEY             ;CONTINUE CHECKING
        BPL COL10       ;

*       PLAYERS TOUCHING PODS?

        DEC PODTME      ;TIME TO CHECK?
        BPL COL13       ;NO! GO AWAY!
        LDA #$03        ;RESET POD TIMER
        STA PODTME      ;TO GO AGAIN.
        LDA P0PF,X      ;SMASHED INTO
        AND #$01        ;POD PLAYFIELD?
        BEQ COL13       ;NO! GO AWAY!
        LDA #$02        ;YES! ROTATE
        JSR RANDO       ;RANDOMLY.
        LDY RANDOM      ;ROTATE RIGHT
        BPL COL12       ;OR LEFT?
        EOR #$FF        ;ROTATE RIGHT.
        CLC             ;(CLOCKWISE)
        ADC #$01        ;
COL12   CLC             ;ROTATE LEFT.
        ADC GEVDIR,X    ;(COUNTERCLOCK)
        JSR WRAP        ;CHECK FOR WRAP
        STA GEVDIR,X    ;AROUND & SAVE
        STA PHASE,X     ;NEW ROTATION.

*       PLAYER HIT BY LASER?

COL13   LDA P0PF,X      ;HAS PLAYER
        AND #$02        ;COLLIDED WITH
        BEQ COLXX       ;LASER PLAYFIELD?
        STX IDIE        ;YES! YES! YES!
        JSR KILLME      ;VAPORIZE HIM!!!
COLXX   DEX             ;CHECK NEXT
        BPL COL5        ;PLAYER...

        STA HITCLR      ;CLEAR COLLISIONS.

        RTS             ;GO TO DARK SIDE OF MOON.

*       INSERT DEATH VALUE

KILLME  STY YHOLD       ;SAVE Y-REG.
        LDY IDIE        ;GET WHO DIES!
        LDA STOP,Y      ;ARE THEY AL-
        BNE :KILLX      ;READY DEAD?
        TXA             ;NO! PREPARE
        TAY             ;TO VAPORIZE!
        JSR ZAPIT       ;ASHES TO ASHES!
        LDY IDIE        ;VAPORIZE OTHER
        JSR ZAPIT       ;VESSEL AS WELL!
        INC NOKILL,X    ;INC # OF KILLS!
:KILLX  LDY YHOLD       ;RESTORE Y-REG.
        RTS             ;GET LOST!!!

*       CHANGE TO DEATH STATUS

ZAPIT   LDA STOP,Y      ;IS THIS VESSEL
        BNE ZAPRTS      ;ALREADY ICED!
        LDA #07         ;NO! START VESSEL
        STA TYPE,Y      ;DETONATION!!!
        LDA #120        ;GIVE THEM A
        STA STOP,Y      ;DEATH STATUS.
        LDA #00         ;TURN OFF BACTER-
        STA LSOUND      ;ION! LASER SOUND.
        LDA #66         ;START UP DETONAT-
        STA XSOUND      ;ION SOUND.
        RTS             ;BEAT IT!
ZAPRTS  PLA             ;PREMATURE
        PLA             ;RETURN. (PACK
        LDY YHOLD       ;YOUR BAGS AND
        RTS             ;HIT THE ROAD!)

IDIE    DB      0       ;VESSEL # TO DIE.
YHOLD   DB      0       ;Y-REG. STORAGE.
PODTME  DB      0       ;ROTATION TIMER.
```

```
 TITLE 'FIRE PLAYER PROJECTILES'

SHOOT   PROC

        JSR FIRST       ;DO THIS FIRST.
        JMP SECOND      ;DO THIS SECOND.

*       INITIALIZE PROJECTILES

FIRST   LDX NOPLAY      ;# OF PLAYERS
SHOOT5  LDA STOP,X      ;IS PLAYER ICED?
        BNE XSHOOT      ;YES! SKIP HIM!
        LDA FDELAY,X    ;OK TO FIRE?
        BEQ SHOOT6      ;YES! CONTINUE.
        DEC FDELAY,X    ;NO! DEC TIMER.
        JMP XSHOOT      ;SKIP TO NEXT.

SHOOT6  LDA NOBULL,X    ;ALL BULLETS
        CMP #$04        ;FIRED ALREADY?
```

ATARI®

600XL . . . . . **CALL**

1200XL . . . . . **CALL***

*Reflects $100 Atari Rebate

1050 DRIVE	**$335**
1025 PRINTER	**$399**
1020 COLOR PTR	**$219**
1027 PRINTER	**$279**
1010 RECORDER	**$72**
MICRO PERIF. MODEM/INFC	

DRIVES

ASTRA 1620	**$469**
RANA 1000	**$295**
PERCOM 88-S1	**$298**
PERCOM 40-S1	**$418**
PERCOM 40-S2	**$718**
PERCOM 44-S1	**$510**
PERCOM 44-S2	**$929**

MICROBITS INFC	**$78**
80 COLUMN BD	**$249**
TECHNICAL NOTES	**$25**
REAL TIME CLK	**$38**
AXIOM PRINTER INFC	**$80**
	$129

MEMORIES

48K RAM (INTEC)	**$95**
64K RAM (INTEC)	**$119**
48K RAM (MOSAIC)	**$109**
64K RAM (MOSAIC)	**$145**
128K RAM DISK	**$299**
32K RAM (MOSAIC)	**$68**

ATARI SOFTWARE

ADVENTURE INT'L
Adv. 1-12 each (C) $18
Preppie (C/D) $20
Preppie II (C/D) $23
Diskey (D) $33
Sea Dragon (C/D) $23
Bug Off! (C/D) $20
Tutti-Frutti (C/D) $17
Saga 1-3 (ea.) (D) $27
Lunar Lndr . (C) $11 (D) $15
Galactic Empire (C) . . $14
Galactic Trader (C) . . . $14
Stratos (C/D) $23
Rear Guard (C) $14 (D) $17

APX
Eastern Front (C/D) . . $23
747 Land Sim. (C/D) . . $17
Fig-Forth (C) $30
Family Cash Flow (D) . $17
Downhill (C/D) $17
Avalanche (C/D) $17
Outlaw/Howitzer (C/D) $17
Salmon Run (C/D) $17
Data Management (D) $17
Galahad/Holy Grail (D) $21
Adv. Music System (D) $21

ATARI INC.
Microsoft Basic II (R) . $62
Mickey in Great Outdoors (C/D) $36
Paint (D) $30
Speed Reading (C) . . . $54
Qix (R) $30
Dig Dug (R) $30
Atari Writer (R) $68
Time Wise (D) $23
Visicalc (D) $139
Juggles House (C/D) . $22
Juggles Rnbw (C/D) . . $22
Pilot (Home) (R) $55
Galaxian (R) $30
Defender (R) $30
ET $34
Microsoft Basic (D) . . $62
Macro Ass. & Edit (D) . $62
Assembler Editor (R) . $42
Basic Cartridge (R) . . . $45
Pac Man (R) $30
Centipede (R) $30
Caverns of Mars (D) . . $28
Star Raiders (R) $30
Conv. Lang. Ea. (C) . . . $42
Music Composer (R) . . $31
Super Breakout (R) . . . $26
My First Alphabet (D) . $26
Prog.2 & E (ea.)(C) . . . $21
Word Processor (D) . $102
Pilot (Educ.) (R) $92
Touch Typing (C) $19
Home File Mngr (D) . . $36
Bookkeeper (D) $102
Family Finance (D) . . . $35
Prog. 1 (C) $18
Scram (C) $18
Asteroids (R) $27
Space Invaders (R) . . . $27
Missile Command (R) . $27
Telelink (R) $21
Superman III (R) $34
Basketball (R) $24
Donkey Kong (R) $34

AVALON HILL
VC (D) $17
B-1 Nuc. Bomber (C) . . $12
Legionnaire (C) $23
Empire of Overmind . . . . . (D) $23 (C) $20
Tanktics . . . (D) $20 (C) $17
Comptr Stock & Bonds . . . . . (D) $17 (C) $14
Dnieper River Line (D) $20
Voyager (D) $17
Controller (D) $20
GFS Sorceress . . . . . (D) $23 (C) $20
Telegard . . (D) $19 (C) $16
Vorrak (D) $18
Lord of Karma (C) $14

AVALON HILL(Cont'd)
Moon Patrol (C) $17
Planet Miners (C) $12
Conflict 2500 (C) $12
Space Station Zulu . . . . . (D) $17 (C) $14
Flying Ace . (D) $21 (C) $18
Gypsy . . . . (D) $18 (C) $15
Galaxy . . . . (D) $17 (C) $14
Guns of Fort Def. (C) . $14
Nuke War (C) $12
Andromeda Conquest . . . . . (D) $16 (C) $13
Close Asslt (D) $23 (C) $20

BRODERBUND
Sky Blazer (D) $22
Bank St. Writer (D) . . . $46
A.E. (D) $23
Arcade Machine (D) . . $39
Choplifter . (D) $23 (R) $29
Labyrinth (C/D) $20
Serpentine . (D) $23 (R) $27
Steller Shuttle (C/D) . . $17
Apple Panic (C/D) $20
Genetic Drift (C/D) . . . $20
David's Midnight (D) . . $23
Sea Fox (D) $20
Track Attack (D) $20
Operatn Whrlwnd (D) . $23
Match Boxes (C/D) . . . $20

CDY CONSULTING
Pogoman (C/D) $27
Mad-Netter (C/D) $23
It-Is-Balloon (C/D) $23

CBS
Mountain King (R) . . . $27
Boulders & Bombs (R) $27
Krazy (each) (R) $27

CONTINENTAL SOFT.
Home Accountant (D) $48
Tax Advantage (D) . . . $39

DATA MOST
Pig Pen (D) $20
Mating Zone (D) $20
Roundabout (D) $20
Bilestoad (D) $20
Night Raiders (D) $20
Monster Smash (D) . . . $20

DATASOFT
Text Wizard (D) $65
Graphic Master (D) . . . $27
Micro Painter (D) $23
Lisp Interpreter (D) . . . $79
Graphics Gen.(D) $17
Basic Compiler (D) . . . $65
Zaxxon (C/D) $27
Teletalk (D) $34
Pac. Coast Hwy (C/D) . $20
Clowns/Balloons (C/D) $20
Spell Wizard (D) $53
Canyon Climber (C/D) $20
Sands of Egypt (D) . . . $27
O'Riley's Mine (C/D) . . $23
Rosen's Brigade (C/D) $23
Moon Shuttle (D) $27

DON'T ASK
Sam (D) $39
P.M. Animator (D) $23
Teletari (D) $27
Poker Sam (D) $17
Word Race . (D) $17 (C) $14

EDU-WARE
Prisoner II (D) $27
Spelling Bee (D) $27
Rendevous (D) $27
Compu/Read . . . . . (D) $21 (C) $15
Compu/Math-FR . . . . . (D) $27 (C) $21
Compu/Math-Dec . . . . . (D) $27 (C) $21

EDUCATIONAL SOFT.
Tricky Tutorial 1,2,3 or 4 (C/D) $15
Tricky Tutorial 5,6 or 7 (C/D) $22

Printers/Etc.

GEMINI 10X	**$279**
GORILLA	**$199**

CITOH

Prowriter	$345
Prowriter II	$629
Starwriter	$1149
Printmaster	$1448

NEC

8023 A-C	$409
3510	$1375
3530	$1579
3550	$1779
7710/7730	$1998

PROWRITER	**$345**
SMITH TPI	**$488**
SILVER REED P	**$669**
QUME 11/40 +	**$1299**

OKI-DATA
Microline 82A . . . . . CALL
Microline 83A . . . . . CALL
Microline 84P . . . . . CALL
Microline 92 . . . . . CALL
Microline 93 . . . . . CALL

DIABLO

620R	$939
630R	$1719

MONITORS

AMDEK

Color I	$289
V300	$139
V300A	$149
Color II	$449

NEC

GRN (JB1260)	$115
GRN (JB1201)	$155
Color Composite	$298
RGB Color	$598

MODEMS

HAYES

Smartmodem	$209
Smartmodem 1200	$498
Micromodem II	$259

NOVATION

J-Cat	$99
Apple Cat II	$259
D-Cat	$149

ANCHOR AUTOMATION
Mark I or II Modem . . . $78

SPECIALS

Gemini 15X Printer	$399
Axiom AT-100 Printer (with interface	$229
Astra Double Density Dual Drive	$469
Rana 1000 Drive	$319
Bit-3 80 Column Board	$245
Mannesmann Talley 160L Printer	$589
Atari 400 Keyboard (In Home)	$35
Programmer Kit . . $48	Entertainer Kit . . . $64
Wico Joystick . . . $23	Wico Trackball . . . $49

COSMIC COMPUTERS
UNLIMITED

727 BREA CANYON RD., SUITE 16
WALNUT, CA 91789
ORDER LINES OPEN MON-SAT 8 am - 8 pm

(800) 626-7642
PLEASE FOR ORDERS ONLY
SORRY, NO COD'S

(714) 594-5204
FOR TECHNICAL INFO, ORDER INQUIRIES, OR FOR CALIFORNIA ORDERS

Add $2.50 shipping per software order in continental U.S. Add $5.00 shipping per software order for AK, HI, FPO-APO. Add $10.00 or 15% (whichever is greater) per software order for non-U.S. Call for cost of hardware shipping. Calif. residents add 6½% sales tax. Cashiers checks or money orders filled within 24 hours for items in stock. Personal checks require 4 weeks to clear. MasterCard and Visa OK for software only within continental U.S., add 3% surcharge. Include card no., expiration date and signature. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call to obtain one before returning goods for replacement or repair. Prices & availability subject to change.

CIRCLE #124 ON READER SERVICE CARD.

*ASTRA 1620
DISK DRIVE SYSTEM

MORE FOR YOUR MONEY
DOUBLE OR SINGLE DENSITY
TWO DRIVES

SPECIAL $469

EPYX-AUTO. SIMULATION
Rescue at Rigel (C/D) $20
Datestones/Ryn (C/D) $14
Upr Rchs Apshai (C/D) $14
Ricochet (C/D) $14
Keys/Acheron (C/D) . . $14
Curse of Ra (C/D) $14
Danger/Drindisti (C/D) $14
Jumpman (C/D) $27
Escp Vulcans Isle (D) . $20
Crypt of Undead (D) . . $20
Nightmare (D) $20
Armor Assault (D) $27
Monster Maze (R) $27
Alien Garden (R) $27
Plattermarna (R) $27
Morloc's Tower (C) . . . $14
King Arthur's Heir . . . . . (D) $20 (C) $14
Hellfire Warrior (C/D) . $27
Invasion Orion (C/D) . . $17
Temple of Aps. (C/D) . $27
Star Warrior (C/D) $27
Dragon's Eye (D) $20
Crush Crumble (C/D) . $20

GEBELLI
Firebird (R) $27
Embargo (R) $30
Candy Factory (D) $20
Match Racers (C/D) . . $20
Dr. Goodcode (D) $20

HAYDEN
Go . . . . . (D) $23 (C) $20
Sargon II . . (D) $23 (C) $20
Bulldog Pinball (C) . . . $20

HES
Coco (C/D) $34
Coco II (C/D) $27

INFOCOM
Suspended (D) $34
Zork I, II or III (D) $27
Starcross (D) $27
Deadline (D) $34
Witness (D) $34

INHOME
Baseball . . (R) $23 (D) $23
Crypts of Terror . . . . . (D) $23 (C) $20

INNOVATIVE DESIGN
Pool 1.5 (D) $23
Juggler (D) $20
Speedway Blast . . . . . (D) $20 (R) $27
Pool 400 (R) $27

JV SOFTWARE
Jrny to Plnts (C/D) $20
Action Quest (C/D) . . . $20
Ghost Encount. (C/D) . $20

LJK
Letter Perfect (D) . . . $104
Data Perfect (D) $74
Letter Perfect (R) . . . $137
Edit 6502 (R) $137

MONARCH DATA SYS.
ABC Compiler (D) $48

ODESTA
Chess (D) $45
Checkers (D) $34
Odin (D) $34

OPTIMIZED SYSTEMS
C-65 (D) $58
Bug-65 (D) $23
Max-65 (D) $58
Basic A + (D) $58
Action (D) $65
Speed Read Plus (D) . . $40

PHOENIX
Birth of Phoenix (D) . . $16
Adv. In Time (D) $20

QUALITY
Fastgammon (C) $14
Name That Song . . . . . (D) $13 (C) $11
Starbase Hyperion . . . . . (D) $17 (C) $14
Ali Baba/40 Thvs (D) . . $20
Jeeper Creepers (D) . . $20

ROKLAN
Gorf . . . . . (D) $27 (R) $30
Wizard/Wor . (D)$27 (R)$30
Dlx Invaders (D)$23 (R)$27
Anti Sub Ptrl (D)$20 (C)$14
Telecom (D) $49
Space Journey (R) . . . $30

SENTIENT
Gold Rush (D) $23
Cyborg (D) $23

SIERRA ON-LINE
Mission Asteroids (D) . $17
Ulys.& Gldn Fleece (D) $27
Crossfire . . (D) $20 (R) $23
Mouseattack (D) $23
Jawbreaker (D) $20 (R) $23
Threshold (D) $27
Softporn (D) $20
Ultima II (D) $39

SIERRA ON-LINE
Marauder (D) $23
Lunar Leeper (D) $20
Wiz & Princess (D) . . . $22
Frogger (C/D) $23

SIRIUS
Alpha Shield (R) $27
Wavy Navy (D) $23
Bandits (D) $23
Space Eggs (D) $20
Sneakers (D) $20
Way Out (D) $27
Type Attack (D) $27
Repton (D) $27
Critical Mass (D) $27
Fast Eddy (R) $23
Worm War (R) $23

SPINNAKER
Snooper Troop 1,2 (D) . $30
Kindercomp (D) $20
Rhymes & Riddles (D) $20
Hey Diddle Diddle (D) . $20
Srch Amzng Thngs (D) $27
Story Machine (D) $23
Face Maker (D) $23

STRATEGIC SIM.
Cosmic Balance (D) . . $27
Cosmic Balance II (D) $27
Tigers In Snow (C/D) . . $27
Battle of Shiloh (C/D) . $27
Battle of Norm. (C/D) . $27
Cytron Masters (D) . . . $27
Knights/Desert (C/D) . $27
Combat Leader (C/D) . $27

SUBLOGIC
Pinball (C/D) $20

SWIFTY
J. White Music Lessons . . . . . (C/D) $20
Space Shuttle (D) $20

SYNAPSE SOFTWARE
File Mngr 800 + ... $65
Protector II (D) $23 (R) $29
Shamus . . . (D) $23 (R) $29
Fort Apocalypse (C/D) $23
Shamus II (C/D) $23
Necromancer (C/D) . . . $23
Pharoh's Curse (C/D) . $23
Slime (C/D) $23
Page 6 (D) $23
Reptilian (C/D) $23
Picnic Paranoia (C/D) . $23
Claim Jumper (C/D) . . $23
Drelbs (C/D) $23
Shadow World (C/D) . . $23
Survivor (C/D) $23
T.N.T. (C/D) $23
N.Y.C. (C/D) $23
Slamball (C/D) $23
Blue Max (C/D) $23
Nautilus (C/D) $23
Chicken (R/D) $23
Quasimodo (C/D) $23
River Quest (C/D) $23

SYNERGISTIC
Programr Wkshp (D) . . $23
Graph Workshop (D) . . $27
Disk Workshop (D) . . . $23
Worlock's Revenge (D) $23

THORN EMI
Soccer (R) $34
Jumbo Jet (R) $34
Submarine Comm. (R) $34
Kick Back (R) $34
Pool (C) $22
Humpty Dumpty (C) . . $22
Hckry Dckry Dck (C) . . $22
Darts (C) $22

USA
Atari World (D) $39
3-D Sprgrphcs (C/D) . . $27
Survival Adv. (C/D) . . . $17

VERSA
Mind Bgglrs I (D)$14 (C)$12
Globe Master (D) $20

MISCELLANEOUS
Financial Wizard (D) . . $41
Castle Wolfenstein (D) $20
Master Type (D) $27
Astro Cha: 'D) $22
Miner 2049e. ') $34
Cypher Bowel (C) $33
Bug Attack (C/D) $20
Raster Blaster (D) $20
Air Strike (C/D) $27
Prism (D) $18
Megalegs (C) $23
Cap'n Cosmo (D) $19
Spy's Demise (D) $14
Galac. Chase (D)$20 (C)$18
Kid Grid (C/D) $20
Battle Trek (D) $20
Snapper (D) $19
T&F-PMP Property Management (D) . . $149
Millionaire (D) $39

```
         BCS XSHOOT   ;YES! SKIP HIM.
         LDA TRIG0,X  ;BUTTON PRESSED?
         BNE XSHOOT   ;NO! SKIP HIM.
         INC NOBULL,X ;INC # BULLETS.
         LDA #$07     ;MAKE CANNON
         STA CSOUND   ;FIRING SOUND.
         LDA #$03     ;SET UP FIRING
         STA FDELAY,X ;DELAY.
         LDY #12      ;FIND AN UNUSED
SHOOT7   LDA BULLET,Y ;ARRAY SLOT FOR
         BMI SHOOT8   ;A PROJECTILE.
         DEY          ;DON'T STOP
         BNE SHOOT7   ;UNTIL YOU DO!!!
SHOOT8   STY BSLOT    ;ARRAY SLOT #.
         TXA          ;SAVE WHICH PLR
         STA BULLET,Y ;# WHO FIRED.
         LDY GEVDIR,X ;GET STARTING
         TYA          ;POINT OF PRO-
         PHA          ;JECTILE ACCORD-
         LDA DELX,Y   ;ING TO PLAYER'S
         ASL A        ;ANGLE OF ROT-
         CLC          ;ATION & (OF
         ADC GEVX,X   ;COURSE)
         SEC          ;PLAYER'S
         SBC #44      ;X & Y COORDS.
         LDY BSLOT    ;
         STA GEVX,Y   ;YUP!
         PLA          ;
         STA GEVDIR,Y ;THAT'S WHAT ALL
         TAY          ;THIS CODE DOES.
         LDA DELY,Y   ;
         ASL A        ;I DARE ANYONE
         CLC          ;TO COME UP WITH
         ADC GEVY,X   ;A BETTER PUBLIC
         SEC          ;DOMAIN PROGRAM.
         SBC #$1C     ;(EXCEPT TOM
         LDY BSLOT    ;HUDSON...)
         STA GEVY,Y   ;SO THERE!!!

XSHOOT   DEX          ;MOVE ALONG TO
         BPL SHOOT5   ;NEXT PLAYER.
         RTS          ;BEGONE!!!

*        MOVE PROJECTILES

SECOND   LDX #12      ;HANDLE ALL.
TRAV5    LDA BULLET,X ;ANYBODY OWN
         BMI XTRAV    ;THIS BULLET?

         JSR MOVSUB   ;YES! MOVE IT!

         LDA GEVX,X   ;OUT OF
         CMP #162     ;BOUNDS?
         BCS :DEACT   ;YES! BYE!
         STA CELLNX+5,X ;
         LDA GEVY,X   ;OUT OF
         CMP #192     ;BOUNDS?
         BCS :DEACT   ;YES! LATER!
         STA CELLNY+5,X ;

         LDY #$04     ;IS THIS BULLET
TRAV6    LDA STOP,Y   ;WITHIN RANGE OF
         BNE TRAV7    ;ANY OF THE
         LDA GEVX,X   ;BACTERIONS!
         CLC          ;
         ADC #48      ;LETHAL RANGE
         SEC          ;IS WITHIN
         SBC GEVX,Y   ;8 UNITS ON
         CMP #$09     ;EITHER X OR Y
         BCS TRAV7    ;AXIS.
         LDA GEVY,X   ;
         CLC          ;
         ADC #$20     ;
         SEC          ;
         SBC GEVY,Y   ;
         CMP #$09     ;
         BCS TRAV7    ;

         JSR ZAPIT    ;EXTERMINATE!!!

         STX :XHOLD       ;SAVE X-REG.
         LDA BULLET,X     ;FIND BULLET
         TAX              ;OWNER & INC.
         INC NOKILL,X     ;# OF KILLS.
         LDX :XHOLD       ;RES. X-REG.

:DEACT   STX :XHOLD       ;SAVE X-REG.
         LDA BULLET,X     ;GET BULLET
         TAX              ;OWNER & DEC.
         DEC NOBULL,X     ;# OF BULLETS.
         LDX :XHOLD       ;RES. X-REG.
         LDA #$FF         ;DEACTIVATE
         STA BULLET,X     ;BULLET. (NO
         STA CELLNX+5,X   ;OWNER OR X,
         STA CELLNY+5,X   ;Y COORDS)
         BNE TRAV8        ;GET OUT!

TRAV7    DEY          ;CHECK FOR COLL-
         CPY #$01     ;ISION WITH NEXT
         BNE TRAV6    ;BACTERION!

TRAV8    LDA #$01         ;BULLET LIVES.
         STA CELLMV+5,X   ;MOVE IT.

XTRAV    DEX          ;MOVE NEXT BULLET.
         CPX #$04     ;OUT OF BULLETS TO
         BNE TRAV5    ;MOVE?
         RTS          ;YES! LATER Y'ALL!

BSLOT    DB       0 ;UNUSED ARRAY SLOT.
:XHOLD   DB       0 ;X-REG. TEMP STORE.

 TITLE 'BACTERION! GRAPHICS PLOTTER'

*        ----------------------
*        GR. 7+ PLOTTER ROUTINE
*        COURTESY OF TOM HUDSON
*        ----------------------

PLOTTER  PROC

*        POINT PLOTTER ROUTINE

PLOTPT   LDX      PLOTY     ;Y-COORD.
         CPX      #193      ;OFFSCREEN?
         BCS      :PBYE     ;YES! QUIT!
         LDA      LOTBL,X   ;NO! GET LO
         STA      LO        ;& HI BYTE
         LDA      HITBL,X   ;OF SCREEN
         STA      HI        ;RAM AREA.
         LDA      PLOTX     ;X-COORD.
         CMP      #160      ;OFFSCREEN?
         BCS      :PBYE     ;YES! QUIT!
         AND      #3        ;PLOT INDEX
         TAX                ;PLACE IN X
         LDA      PLOTX     ;GET PLOTX &
         LSR      A         ;DIVIDE
         LSR      A         ;BY 4.
         STA      YOFSET    ;
         LDY      COLOR     ;GET COLOR &
         LDA      :BMSK2,X  ;MASK OFF
         AND      :COLRS,Y  ;PIXEL POS.
         STA      HOLD      ;SAVE IT.
         LDA      :BMSK1,X  ;MASK OFF PIXEL
         LDY      YOFSET    ;OF ADDR TO BE
         AND      (LO),Y    ;ALTERED.
         ORA      HOLD      ;SET PLOT BITS
         STA      (LO),Y    ;& STORE.
:PBYE    RTS                ;ALL DONE...

*        DRAW FROM/TO ROUTINE

DRAWTO   LDA      DRAWY     ;IS DRAWY
         CMP      PLOTY     ;> PLOTY?
         BCC      :YMNUS    ;NO!
         SEC                ;SUB. PLOTY
         SBC      PLOTY     ;FROM DRAWY &
         STA      DELTAY    ;SAVE DIFF.
         LDA      #1        ;Y INC. = 1.
         STA      INCY      ;(DOWN)
         BNE      :XVEC     ;BRANCH!
:YMNUS   LDA      PLOTY     ;SUB. DRAWY
         SEC                ;FROM PLOTY &
         SBC      DRAWY     ;SAVE DIFF.
         STA      DELTAY    ;
         LDA      #255      ;Y INC. = -1.
         STA      INCY      ;(UP)
:XVEC    LDA      DRAWX     ;IS DRAWX
         CMP      PLOTX     ;> PLOTX?
         BCC      :XMNUS    ;NO!
         SEC                ;SUB. DRAWX
         SBC      PLOTX     ;FROM PLOTX
         STA      DELTAX    ;& SAVE DIFF.
         LDA      #1        ;X INC. IS 1
         STA      INCX      ;(RIGHT)
         BNE      :VCSET    ;BRANCH!
:XMNUS   LDA      PLOTX     ;SUB. DRAWX
         SEC                ;FROM PLOTX
         SBC      DRAWX     ;& SAVE DIFF.
         STA      DELTAX    ;
         LDA      #255      ;X INC IS -1
         STA      INCX      ;(LEFT)
:VCSET   LDA      #0        ;ZERO OUT
         STA      ACCY      ;Y-ACC.
         STA      ACCX      ;X-ACC.
         LDA      DELTAX    ;IS DELTAX >
         CMP      DELTAY    ;DELTAY?
         BCC      :YMAX     ;NO!
         STA      COUNTR    ;SAVE DELTAX
         STA      ENDPT     ;IN COUNTR. ENDPT.
         LSR      A         ;DIV. BY 2 &
         STA      ACCY      ;STORE IN Y-ACC.
         JMP      :DRAW8    ;START DRAW
:YMAX    LDA      DELTAY    ;DELTAY LARGER.
         STA      COUNTR    ;STORE IT IN
         STA      ENDPT     ;COUNTR, ENDPT.
         LSR      A         ;DIV BY 2 &
         STA      ACCX      ;STORE IN X-ACC.

*        BEGIN DRAWING TO DESTINATION

:DRAW8   LDA      COUNTR    ;IF COUNTR=0...
         BEQ      :DRAWE    ;NO DRAW!
:BEGIN   LDA      ACCY      ;ADD DELTAY
         CLC                ;TO Y-ACC.
         ADC      DELTAY    ;
         BCS      :OVER1    ;
         STA      ACCY      ;
         CMP      ENDPT     ;AT ENDPNT YET?
         BCC      :BEGN2    ;NO, GO DO X.
         LDA      ACCY      ;SUB. ENDPT
         SEC                ;FROM Y-ACC.
:OVER1   SBC      ENDPT     ;
         STA      ACCY      ;
         LDA      PLOTY     ;AND INC. THE
         CLC                ;Y. POSITION.
         ADC      INCY      ;
         STA      PLOTY     ;
:BEGN2   LDA      ACCX      ;ADD DELTAX TO
         CLC                ;X-ACC.
         ADC      DELTAX    ;
         BCS      :OVER2    ;
         STA      ACCX      ;
         CMP      ENDPT     ;AT ENDPT. YET?
         BCC      :PLOTT    ;NO, GO PLOT.
         LDA      ACCX      ;SUB. ENDPT
         SEC                ;FROM X-ACC.
:OVER2   SBC      ENDPT     ;
         STA      ACCX      ;
         LDA      PLOTX     ;AND INC.
         CLC                ;PLOT X
         ADC      INCX      ;
         STA      PLOTX     ;
:PLOTT   JSR      PLOTPT    ;PLOT POINT.
         DEC      COUNTR    ;MORE TO DRAW?
         BNE      :BEGIN    ;YES!
:DRAWE   RTS                ;NO, ALL DONE...

*        DRAW AN INDIVIDUAL CELL

SHOCEL   LDX CELNUM    ;CELL # TO DRAW
         LDA #$00      ;SPECIFY COLOR
         STA CELLMV,X  ;& DON'T UPDATE
         STA COLOR     ;AGAIN.
         LDA CELLOX,X  ;GET OLD X-COORD
         STA PLOTX     ;& STORE
         LDA CELLOY,X  ;GET OLD Y-COORD
         STA PLOTY     ;& STORE
         LDA #$00      ;SPECIFY OBJECT
         JSR OBJECT    ;& DRAW IT...
         LDX CELNUM    ;CELL # TO DRAW
         LDA CELLNX,X  ;GET NEW X-COORD
         STA PLOTX     ;& STORE
         STA CELLOX,X  ;
         LDA CELLNY,X  ;GET NEW Y-COORD
         STA PLOTY     ;& STORE
         STA CELLOY,X  ;
         LDA #1        ;SPECIFY COLOR
         STA COLOR     ;OF CELL
         LDA #0        ;SPECIFY OBJECT
         JSR OBJECT    ;& DRAW IT...
         RTS           ;ALL DONE...

*        CELL MOVER

OBJECT   ASL      A         ;MULT OBJECT
         ASL      A         ;INDEX BY 8
         ASL      A         ;TO POINT INTO
         STA      SHAPIX    ;SHAPE TABLE
         LDA      #8        ;8 LINES MAX
         STA      SHAPCT    ;IN SHAPE
DOBLP    LDX      SHAPIX    ;GET LINE #
         LDY      OBJDIR,X  ;& ITS DIRECTION
         BMI      ENDOBJ    ;IF $FF ALL DONE
         LDA      PXINC,Y   ;GET X INCREMENT
         STA      XI        ;
         LDA      PYINC,Y   ;AND Y INCREMENT
         STA      YI        ;
         LDA      OBJLEN,X  ;AND LINE LENGTH
         STA      LENGTH    ;
PLOTOB   LDA      PLOTX     ;INCREMENT
         CLC                ;THE X
         ADC      XI        ;COORDINATE
         STA      PLOTX     ;AND SAVE
         LDA      PLOTY     ;INCREMENT
         CLC                ;THE Y
         ADC      YI        ;COORDINATE
         STA      PLOTY     ;AND SAVE
         LDA      SHAPCT    ;FIRST LINE?
         CMP      #$08      ;
         BEQ      NOPLT1    ;DON'T PLOT IT!
         JSR      PLOTPT    ;PLOT POINT
NOPLT1   LDA      LENGTH    ;MORE LENGTH?
         BEQ      NOOBJ     ;
         DEC      LENGTH    ;DECREASE LENGTH
         BNE      PLOTOB    ;YUP!
NOOBJ    INC      SHAPIX    ;NEXT LINE
         DEC      SHAPCT    ;DONE 8 LINES?
         BNE      DOBLP     ;NOPE!
ENDOBJ   RTS                ;FINIS!!!

*        SHAPE DATA

PXINC    DB       0,0,1,$FF,1,$FF,1,$FF
PYINC    DB       $FF,1,0,0,$FF,1,1,$FF

*        OBJECT SIDE LENGTHS

OBJLEN   DB       3,2,2,2,2,2,2,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0
         DB       0,0,0,0,0,0,0,0

*        OBJECT SIDE DIRECTIONS

OBJDIR   DB       2,5,3,7,4,2,6,$FF
         DB       0,0,$FF,0,0,0,0,0
         DB       0,1,$FF,0,0,0,0,0
         DB       0,2,$FF,0,0,0,0,0
         DB       0,3,$FF,0,0,0,0,0
         DB       0,4,$FF,0,0,0,0,0
         DB       0,5,$FF,0,0,0,0,0
         DB       0,6,$FF,0,0,0,0,0
         DB       0,7,$FF,0,0,0,0,0

:COLRS   DB       $00,$55,$AA,$FF
:BMSK1   DB       $3F,$CF,$F3,$FC
:BMSK2   DB       $C0,$30,$0C,$03
:COLR1   DB       $40,$10,$04,$01
LOTBL    DS 192
HITBL    DS 192
         END $2800
```

●

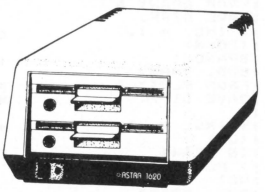
FOR
ALL
ATARI COMPUTERS

BEST SELLERS FROM THE PROGRAMMERS WORKSHOP

TWO DRIVES FOR THE PRICE OF ONE

$469.00

THE ASTRA 1620



MORE DISK DRIVE FOR YOUR MONEY ...

In fact, with the ASTRA 1620, you get two superb Disk Drives for the price of one. The ASTRA 1620 is Single or Double Density (software selectable) and completely compatible with ATARI DOS or OSA+DOS. When used as Double Density, the ASTRA 1620 has the same capacity as Four ATARI 810® Disk Drives.

★ *Satisfaction Guaranteed* ★

INCLUDED: **at no extra charge**

One Copy of Home Writer and One Copy of Smart DOS.

THE HOME WRITER $39.00

The HOME WRITER is an easy to use word processor which includes a carefully selected group of functions that are at your disposal immediately. The functions are as follows: SAVE, LOAD, REVIEW, PRINTOUT, or EDIT. All the popular editting features available on the ATARI Home Computer in direct programming mode are also available with HOME WRITER. You may also search for phrases or strings, load off of either disk or cassette, and number pages automatically. Unlike other small word processing programs, HOME WRITER does not wrap-around when at the end of a line. Right and left margin justification is available for any type parallel printer. 48K.

FILING SYSTEM $39.00

FILING SYSTEM allows the user to configure any type of data file imaginable. Examples are recipe cards, mail lists, reminders for birthdays, check-ups, etc..., complete inventories (home and business), personnel files, customer call-ups, price list, and much, much more. You may retrieve data using any field or combination of fields. Files also may be saved, sorted, and printed in a preset format that you configure. Uses either a single or a double density disk drive. Three preconfigured files are included – a Mailing List, a Price List, and a Household Inventory. 24K minimum. Now in double density.

THE PROGRAMMERS WORKSHOP

5230 Clark Ave., Suite 19

Lakewood, CA 90712

(213) 920-8809

DESK SET $39.00

DESK SET is a perpetual calendar, an appointment calendar and also a card file. The perpetual calendar is a calendar of every month, past, present or future. The appointment calendar allows up to 15 entries to be made each day. The card file is a mail list program which holds up to 200 addresses. The printing format of card file includes continuous lists, labels or envelopes. Files can be printed; all the files from one file number to another; by zip code; by state or by selected files. DESK SET is an easy way to organize your life. 40K

FINANCIAL CALCULATOR $29.00

The program answers virtually any questions concerning the cost of money, loans, and interest earned on savings, loans and investments. Plus, this program will give a complete interest earned table and amortization table. This program is a must for anyone serious about money. 32K.

FORECASTER $29.00

Forecast future events based on past information. Forecast profits, costs, sales trends, prices, test scores, virtually anything. Edit, save on disk and test various elements to determine the outcome. FORECASTER is a powerful "what if" program – a must for business. 24K

STAT PLUS $29.00

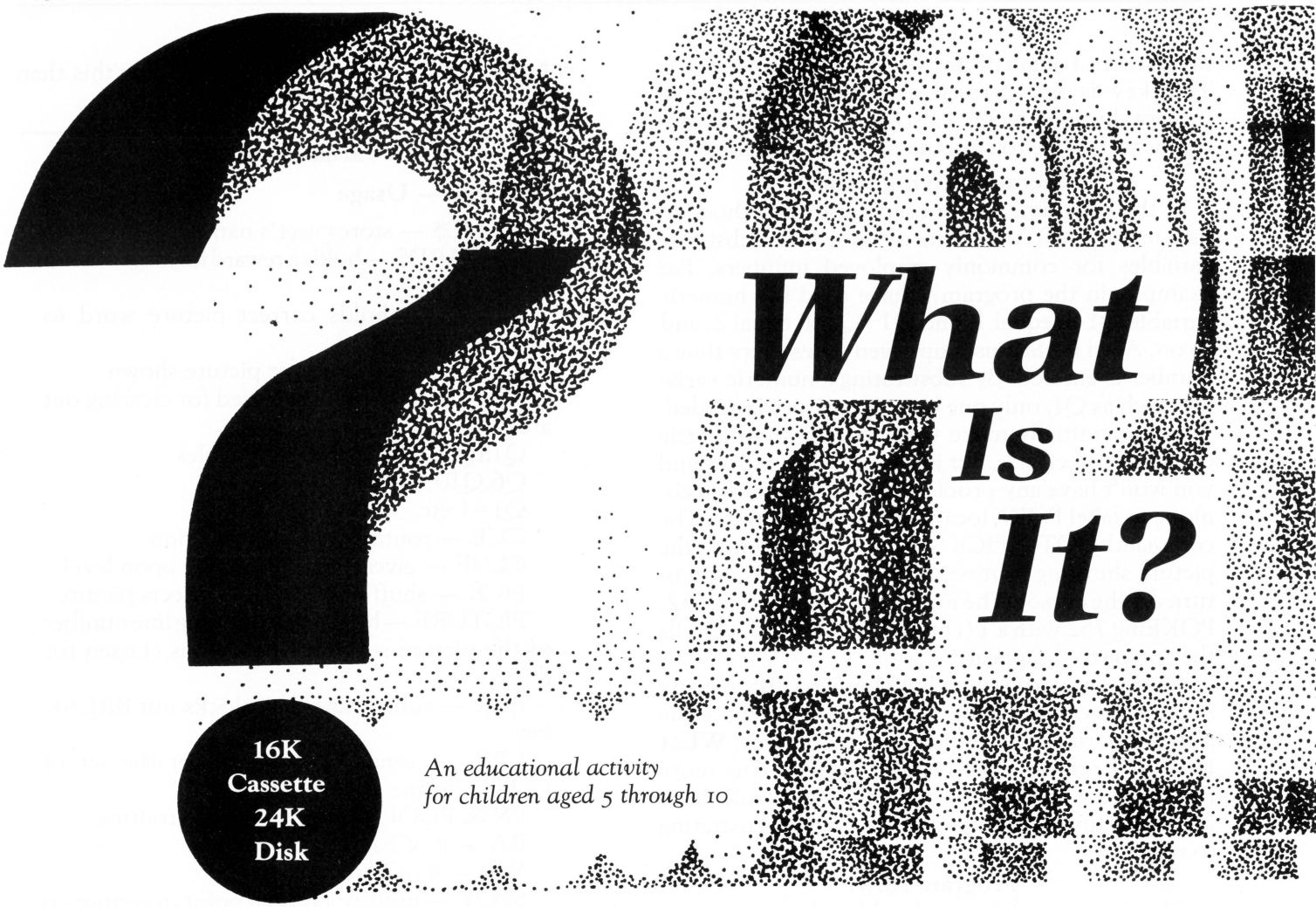
STAT PLUS is the most powerful statistic tool we have seen. Run sample space, mean, variance and standard deviation. Do probabilities using binomial, poisson or normal distribution. Also, do students' t-test, Mann-Whitney U test and Chi Square. Do linear regressions; may use 1-12 independent variables. Interact a sample from one module to another, edit, rank and print out. 24K min.

DISK FIX KIT $29.00

It takes more than a speed adjustment to properly set up an Atari® 810 Disk Drive. Do it Right – Test Disk, Cleaner, Tools, Special Oil, Swabs and complete instructions.

TO ORDER: **VISA/MasterCard, check or money order accepted. If charge, please include expiration date of card. Shipping and Handling software $1.50, disk drive, $10.00. California residents add sales tax. Phone or mail.**

® ATARI is a registered trademark of Warner Communications



by Larry W. Linson

What Is It? is an activity that I began writing for my first grade class, in the fall of 1982. It's a reading/spelling activity in which you are asked to identify a series of seven random pictures. The program is similar to the reading workbooks children use in the primary grades. The child simply needs to type in the picture-word that corresponds with the picture. For example, if a picture of a HOUSE appears, then the word "HOUSE" should be typed in. In **What Is It?** there are four levels of difficulty. Level 1 offers a single clue, the name of the picture. Level 2 gives the picture word and two distracting clues which are not similar to the picture word. Level 3 gives the picture word and three other similar distractors such as HOUSE, HORSE, MOUSE and HOWLS. Level 4 gives no hints at all; the word must be typed in from the child's memory. The program keeps score and reports it after each correct answer. The program keeps score and reports it after each correct answer. I've now had close to fifty children test this program, and most of the "bugs" have been worked out, with many new features added to the original program.

Error trapping.

Children working on Atari computers have the most trouble with accidentally pressing the BREAK key when they want the BACK SPACE key, which is right "next door." I avoided this problem by OPEN-ing the keyboard, rather than using the INPUT command. By using this technique, if a mistake is made and BACK SPACE is pressed, then the program clears the student's response and waits for another. An undesired response can be cleared at any time by pressing either the SPACE BAR or the BACK SPACE key.

The BREAK key was disabled to prevent the program from being stopped prematurely. This was accomplished by using POKE 16,64 and POKE 53774,64. I used a short subroutine at Line 13000 for this, since the POKE must be repeated after each graphics mode change. I also used various TRAP statements for "catching" errors. In this way, the program doesn't halt if an error is detected. The children in my classes have been rather *inventive* in finding ways to "break" or "crash" programs. To date, **What Is It?** hasn't been crashed by any of my

first graders. I didn't protect against SYSTEM RESET. This key is not pressed accidentally very often (except during games!). I also felt it might be advantagous to be able to stop the program if desired.

Memory-saving tricks.

In the program, I have used various methods to save memory (RAM). One of these is to substitute variables for commonly employed numbers. For example, in the program, I have used the numeric variable Q1 to equal 1, and Q1+Q1 to equal 2, and so on. Atari BASIC uses up seven bytes every time a number is entered. By substituting a numeric variable, such as Q1, only one byte of memory is needed. This may initially make the program seem a little difficult to decipher, but just read "Q1" as "1" and you won't have any problem. I used this same technique to label POKE locations and subroutines. The command GOTO PICK sends the program to the picture shuffling routine. POKE OFF,Q1 is used to turn off the cursor. The number held by OFF is 752. POKEing 752 with a 1 (entered as POKE 752,1) tells the cursor to disappear.

Another way of saving RAM was to utilize many statements on the same line, employing the same numeric variables over and over. In this way, **What Is It?** will run on a 16K 400 or 600XL. Any more RAM-saving stunts, and the program would have been virtually unreadable and much more frustrating to type in!

Program flow.

The program is set up rather logically—to my way of thinking, anyway! **What Is It?** begins with a title page and then asks for your name. You select the level you wish to try, and a series of seven random pictures follows. Having only eleven pictures in the program's library may not seem like very many, but I believe that my students enjoy the familiarity and reinforcement they achieve with this library of pictures. Since each picture is picked at random, and the program will not pick the same picture twice, there are over one and one-half million different combinations of the eleven pictures! Check it yourself — try this on your Atari: PRINT (11*10*9*8*7*6*5). The results are staggering, aren't they?

After each picture word is correctly answered, a short reward sequence is initiated. After the seventh picture, an overall score is given, and you are asked if you would like to try again. Many of the techniques that I have used in **What Is It?** have been borrowed from other programmers. The idea of using numeric variables comes from Jerry White; the picture-shuffling routine in Lines 4000-4010 is from James Korenthal; and the practice of using DATA statements to READ numbers to PLOT and DRAWTO for graphics, I learned from Elaine Garringer. I have found that the best way to learn how to program is to type in programs, such as this one, RUN it and then study the code to see how the author achieved different effects. I've learned more from doing this than from any book I have purchased! □

Variable table.

Variable — Usage

NAME$ — stores user's name
REWARD$ — holds a reward message for end of program
TEMP$ — holds correct picture word to match to answer
G$ — user's answer for picture shown
T$ — large blank space used for clearing out area
Q1,Q2,Q3 — numeric variables
Q6,Q10,Q20
Q1=1 etc.
CUE — routine that asks question
CLUE — gives clues depending upon level
PICK — shuffle routine that selects picture
PICTURE — holds the beginning line number of the picture subroutine that was chosen for display
BRK — subroutine that blocks out BREAK key
LWL — sends program to get the set of CLUEs for the appropriate level
PN & POOL — used in shuffle routine
RA — # of Right Answers
WA — # of Wrong Answers
SPOT — used as a return point in setting up TRAP statement
LV — the selected level
HOU,STA,BOX,KIT,BOO,TV,TRU,FOOT, FAC,CON,LOL — used in subroutine that weeds out pictures already used, so the same picture is not shown twice
COUNT — Keeps track of the number of pictures shown.
OFF — stores POKE location 752, turns off cursor
KOLOR — holds the RANDOM number used to POKE into locations 710 & 712, which control the background and text window colors.

The rest of the variables are used as simple counters, for delay statements or PLOT and DRAWTO routines.

Take-apart.

Lines 0-5 — Sends program to initialize variables at Line 15000.

Lines 6-19 — Displays title page and gets child's name.

Lines 20-35 — Child selects level of difficulty and program assigns variables for the appropriate level.

Lines 40-90 — The routine that draws the pictures and stores the correct answer for the

computer to compare to the child's response.

Line 95 — Sends program to the subroutine to display clues assigned to that level, and then on to the input routine.

Lines 100-1110 — DATA for the eleven pictures.

Lines 1500-1512 — The answer INPUT routine.

Lines 1950-1960 — Reward sequence.

Line 1962 — Selects random number for background color and text window.

Line 1963 — Sends program to randomly select a new picture.

Lines 2000-2005 — Incorrect answer sequence, sends program back for clues so child can enter the correct answer.

Lines 4000-4030 — Sequence to randomly select a picture and check to see if picture has already been used, if so program goes back to select another.

Lines 4060-4079 — Ending sequence — based on performance, selects an overall rating and asks if child would like to try again.

Lines 5000-5100 — Clues for Level 1.

Lines 6000-6100 — Clues for Level 2.

Lines 7000-7100 — Clues for Level 3.

Line 8000 — Level 4 offers no clues, so program is returned to answer input routine.

Line 13000 — Routine to POKE out BREAK key, which must be done after each graphics mode change.

Line 13500 — TRAPS keyboard errors and returns program to the proper SPOT.

Line 14000 — Sets graphics mode and screen color for pictures, turns off cursor.

```
0 REM WHAT IS IT? LW.LINSON
5 GOSUB 15000:GRAPHICS Q2:GOSUB BRK:PO
KE 710,48:POKE 712,48:POKE 708,26:RA=Q
0:WA=Q0:POKE OFF,Q1
6 FOR D=Q2 TO 16 STEP Q2:POSITION D,Q0
:? #Q6;"◣":POSITION D,Q2:? #Q6;"◣":N
EXT D
7 POSITION Q2,Q1:? #Q6;"◣ WHAT IS IT?
 ◥"
9 POSITION Q5,Q6:? #Q6;"◢c ┼ ┴●":POSI
TION Q2,Q9:? #Q6;"Larry W. Linson":SPO
T=17:TRAP QTRAP
11 FOR D=Q1 TO 1000:NEXT D:POSITION Q6
,Q9:? #Q6;T$:POSITION Q2,Q9:? #Q6;"    
<"
12 POSITION Q3,Q4:? #Q6;"what is":POSI
TION Q5,Q6:? #Q6;" your name?"
13 POKE OFF,Q1:? :? :? "       [SPACE B
AR]  ERASES NAME"
14 POKE 755,Q2:POKE KEY,Q255:OPEN #Q1,
Q4,Q0,"K:":POSITION Q7,Q9:FOR D=Q1 TO
12:POKE 702,64:POKE 694,Q0
15 GET #Q1,A:IF A=Q155 THEN 19
16 IF A<Q65 OR A>Q90 THEN A=Q32
17 IF A=Q32 OR A=Q126 THEN POSITION Q7
,Q9:? #Q6;T$:CLOSE #Q1:NAME$="":TRAP Q
TRAP:GOTO 14
18 ? #Q6;CHR$(A);:NAME$(LEN(NAME$)+Q1)
=CHR$(A):NEXT D
19 IF D>12 OR NAME$="" THEN A=Q32:GOTO
 17
20 FOR D=Q1 TO Q52:NEXT D:POKE OFF,Q1:
CLOSE #Q1:GOSUB 16000
27 TRAP Q29:CLOSE #Q1
28 POKE KEY,Q255:POKE OFF,Q1:? :? "Wha
t level, ";NAME$;"?";"   1  2  3  4"
29 POKE 702,64:POKE 694,Q0:CLOSE #1:GO
SUB BRK:OPEN #Q1,Q4,Q0,"K:":GET #Q1,L:
X0=6
30 IF L=Q49 THEN L=Q0:LV=Q1:CLOSE #Q1:
GOTO PICK
31 IF L=50 THEN L=1000:LV=Q2:CLOSE #Q1
:GOTO PICK
32 IF L=51 THEN L=2000:LV=Q3:CLOSE #Q1
:GOTO PICK
33 IF L=Q52 THEN L=3000:LV=Q4:CLOSE #Q
1:GOTO PICK
34 IF L<Q49 OR L>Q52 THEN GOTO Q29
35 IF L<Q49 OR L>Q52 THEN GOTO Q29
40 GOSUB KOLOR
50 READ A,B,C,D,E,F,G
60 PLOT A,B:DRAWTO C,D:DRAWTO E,F:IF G
=-Q1 THEN TEMP$="HOUSE":G=Q0:HOU=99:GO
TO 110
61 IF G=-Q2 THEN G=Q10:TEMP$="STAR":ST
A=199:GOTO Q90
62 IF G=-Q3 THEN G=Q20:TEMP$="BOX":BOX
=299:GOTO Q90
63 IF G=-Q4 THEN G=30:TEMP$="KITE":KIT
=399:GOTO Q90
65 IF G=-Q5 THEN G=50:TEMP$="BOOK":BOO
=599:GOTO Q90
66 IF G=-Q6 THEN G=60:TEMP$="TV":TV=69
9:GOTO Q90
68 IF G=-Q7 THEN G=80:TEMP$="TRUCK":TR
U=899:GOTO 915
69 IF G=-Q8 THEN G=90:TEMP$="FOOTBALL"
:FOOT=999:GOTO 1020
70 IF G=-Q9 THEN G=100:TEMP$="FACE":FA
C=1099:GOTO Q90
80 GOTO 50
90 IF LV=Q4 THEN G=Q0
95 GOSUB CLUE+L+G:GOTO CUE
99 RESTORE 100:GOTO Q40
100 DATA 50,30,80,15,110,30,0,110,30,5
0,30,50,70,0,50,70,110,70,110,30,0,110
,30,130,20,130,60,0
101 DATA 130,60,110,70,110,70,0,105,5,
130,20,130,20,0,80,15,105,5,105,5,0
102 DATA 75,16,75,5,65,10,0,65,10,65,2
1,65,21,0,60,24,60,10,71,4,0,71,4,75,4
,75,4,0,65,10,60,10,60,10,0
103 DATA 75,55,75,70,85,70,0,85,70,85,
55,75,55,0,70,50,58,50,58,40,0,58,40,7
0,40,70,50,0,64,49,64,41,64,41,0
104 DATA 59,45,69,45,69,45,0,90,40,90,
50,102,50,0,102,50,102,40,90,40,0,91,4
5,101,45,101,45,0,96,41,96,49,96,49,-1
110 READ H,I,J,K,L1
115 PLOT H,I:PLOT J,K:IF L1=-Q1 THEN G
OTO Q90
116 GOTO 110
120 DATA 61,4,59,3,0,57,2,68,2,0,66,1,
66,4,0,64,3,62,2,0,55,1,52,1,0,60,1,58
,0,0,64,0,83,63,-1
199 RESTORE 200:GOTO Q40
200 DATA 83,8,106,66,50,32,1,50,32,116
,32,60,66,1,60,66,83,8,83,8,-2
299 RESTORE 300:GOTO Q40
300 DATA 100,20,80,10,60,20,0,60,20,80
,30,100,20,0,100,20,100,50,80,60,0
301 DATA 80,60,60,50,60,20,0,80,30,80,
60,80,60,-3
399 RESTORE 400:GOTO Q40
400 DATA 60,10,40,30,60,62,0,60,62,80,
30,60,10,0,60,62,64,70,73,73,0,73,73,7
7,65,85,68,0,85,68,90,66,95,70,0
401 DATA 95,70,103,65,103,65,0,60,11,6
0,61,60,61,0,41,30,78,30,78,30,-4
499 G=Q40:GOSUB KOLOR:FOR W=50 TO 102:
X=Q7:Y=76:Z=61
502 PLOT W,X:DRAWTO Y,Z:NEXT W:TEMP$="
CONE":CON=499:POKE OFF,Q1:GOTO Q90
599 RESTORE 600:GOTO Q40
600 DATA 102,70,62,70,62,22,0,62,22,10
2,22,102,70,0,102,70,108,64,108,16,0,1
08,16,68,16,62,22,0
601 DATA 108,16,102,22,102,22,0,66,18,
106,18,106,66,0,64,20,104,20,104,68,0,
76,28,72,28,72,34,0
```

```
602 DATA 74,32,72,32,72,32,0,80,34,80,
28,84,28,0,84,28,84,34,80,34,0,88,28,9
2,34,92,34,0,88,34,92,28,92,28,0
603 DATA 76,42,76,48,76,48,0,80,48,80,
42,84,48,0,84,48,84,42,84,42,0,68,56,6
4,56,68,62,0,68,62,64,62,64,62,0
604 DATA 72,62,72,56,76,56,0,76,56,76,
62,72,62,0,84,56,80,56,80,62,0,80,62,8
4,62,84,62,0,88,56,88,62,88,62,0
605 DATA 92,56,88,60,92,62,0,100,56,96
,56,100,62,0,100,62,96,62,96,62,-5
699 RESTORE 700:GOTO Q40
700 DATA 112,70,50,70,50,30,0,50,30,11
2,30,112,70,0,112,70,120,62,120,22,0,1
20,22,58,22,50,30,0
701 DATA 120,22,112,30,112,30,0,58,36,
58,66,102,66,0,102,66,102,36,58,36,0,8
8,24,84,24,84,26,0,84,26,88,26,88,24,0
702 DATA 88,24,102,6,102,6,0,84,24,72,
6,72,6,0,108,38,106,38,106,36,0,106,36
,108,36,108,38,0
704 DATA 108,44,106,44,106,42,0,106,42
,108,42,108,44,0,108,50,106,50,106,48,
0,106,48,108,48,108,50,-6
799 GOSUB KOLOR:RESTORE 804
802 READ A,B,C,D
803 PLOT A,B:DRAWTO C,D:IF D=0 THEN GO
TO 850
804 DATA 90,30,90,38,92,22,92,44,94,20
,94,48,96,18,96,50,98,14,98,52,100,14,
100,54,102,12,102,56
805 DATA 104,10,104,57,106,9,106,58,10
8,9,108,59,110,8,110,59,112,7,112,60,1
14,7,114,60,116,6,116,61
806 DATA 118,7,118,60,120,7,120,60,122
,8,122,59,124,9,124,59,126,9,126,58,12
8,10,128,57,130,12,130,56
807 DATA 132,14,132,54,134,14,134,52,1
36,18,136,50,138,20,138,48,140,22,140,
44,142,30,142,38,30,62,32,66
808 DATA 32,66,93,47,91,45,30,62,0,0,0
,0
849 IF D>Q1 THEN GOTO 802
850 G=70:TEMP$="LOLLIPOP":LOL=799:POKE
 752,Q1:GOTO Q90
899 RESTORE 900:GOTO Q40
900 DATA 76,12,58,12,52,18,0,52,18,52,
58,76,58,0,76,58,76,32,76,12,0,76,32,5
2,32,52,32,0
901 DATA 81,33,81,13,77,9,0,77,9,57,9,
49,17,0,49,17,49,33,21,33,0,21,33,17,3
6,17,39,0,17,51,21,49,21,43,0
902 DATA 21,43,17,39,15,43,0,15,43,15,
47,17,51,0,17,51,17,59,23,65,0,23,65,3
1,65,31,55,0,31,55,36,52,44,52,0
903 DATA 44,52,47,55,47,65,0,47,65,101
,65,101,55,0,101,55,106,52,114,52,0,11
4,52,117,55,117,65,0
904 DATA 117,65,137,65,144,58,0,144,58
,144,36,141,33,0,141,33,81,33,81,33,0,
34,66,36,70,44,70,0
905 DATA 44,70,46,66,46,66,0,104,66,10
6,70,114,70,0,114,70,116,66,116,66,-7
915 X=55:Y=X:X1=X:Y1=X
916 PLOT 131,X:DRAWTO 147,Y:PLOT 14,X1
:DRAWTO 29,Y1:X1=X1+Q1:Y1=Y1+Q1:X=X+Q1
:Y=Y+Q1:IF Y1=62 THEN GOTO 940
919 GOTO 916
940 X=38:Y=X
941 PLOT 23,X:DRAWTO 140,Y:X=X+Q1:Y=X:
IF Y=41 THEN PLOT 24,43:DRAWTO 140,43:
GOTO 950
944 GOTO 941
950 X=58:Y=X:X1=X:Y1=Y
951 PLOT 38,X:DRAWTO 42,Y:PLOT 108,X1:
DRAWTO 112,Y1:X1=X1+Q1:Y1=X1:X=X1:Y=Y1
:IF Y1=62 THEN GOTO Q90
955 GOTO 951
999 RESTORE 1000:GOTO Q40
1000 DATA 20,40,24,46,30,54,0,30,54,40
,60,50,64,0,50,64,58,66,72,70,0,72,70,
88,70,102,66,0,102,66,110,64,120,60,0
1001 DATA 120,60,130,54,136,46,0,136,4
6,140,40,136,34,0,136,34,130,26,120,20
,0,120,20,110,16,102,14,0
1002 DATA 102,14,94,12,80,10,0,80,10,7
2,10,58,14,0,58,14,50,16,40,20,0,40,20
,30,26,24,34,0,24,34,20,40,20,40,0
1003 DATA 40,60,50,61,50,61,0,40,60,50
,62,50,62,0,40,60,50,63,50,63,0,40,20,
50,19,50,19,0,40,20,50,18,50,18,0
1004 DATA 40,20,50,17,50,17,0,120,60,1
10,63,110,63,0,120,60,110,62,110,62,0,
120,60,110,61,110,61,0
1005 DATA 120,20,110,19,110,19,0,120,2
0,110,18,110,18,0,120,20,110,17,110,17
,-8
1020 X=34:Y=X
1022 PLOT 66,X:DRAWTO 96,Y:X=X+Q2:Y=X:
IF Y=44 THEN GOTO 1030
1028 GOTO 1022
1030 X=72:Y=X
1032 PLOT X,Q32:DRAWTO Y,44:X=X+Q3:Y=X
:IF Y>92 THEN 1040
1038 GOTO 1032
1040 X=Q20:Y=X:X1=X:Y1=Y
1042 PLOT Q40,X:DRAWTO 50,Y:PLOT 110,X
1:DRAWTO 120,Y1:X1=X1+Q1:Y1=X1:X=X1:Y=
Y1:IF Y1=62 THEN GOTO Q90
1048 GOTO 1042
1099 GOSUB 1107:RESTORE 1100:GOTO 50
1100 DATA 65,13,97,13,105,25,1,105,25,
105,57,97,65,1,97,65,65,65,55,57,1,55,
57,55,25,65,13,1,67,27,73,27,73,33,1
1102 DATA 73,33,67,33,67,27,1,89,27,95
,27,95,33,1,95,33,89,33,89,27,1,65,49,
69,57,73,59,1,73,59,89,59,93,55,1
1104 DATA 93,55,97,49,89,51,1,89,51,73
,51,65,49,1,89,55,89,55,73,55,1,65,49,
73,55,73,55,1,65,49,73,55,73,55,1
1105 DATA 97,49,89,55,89,55,1,79,35,79
,43,83,43,1,83,43,83,45,77,45,1,77,45,
77,35,79,35,1,73,29,69,29,69,33,1
1106 DATA 89,29,93,29,93,33,-9
1107 GOSUB KOLOR:X=73:Y=X
1108 PLOT X,51:DRAWTO Y,59:X=X+Q4:Y=X:
IF Y=93 THEN RETURN
1110 GOTO 1108
1500 POKE KEY,Q255:SOUND Q0,Q40,Q10,14
:FOR D=Q1 TO Q20:NEXT D:SOUND Q0,Q0,Q0
,Q0:? :? :? "      This is a ...";
1501 SPOT=1507:TRAP QTRAP:OPEN #Q1,Q4,
Q0,"K:":GOSUB BRK
1505 FOR D=Q1 TO 12:GET #Q1,A:IF A=Q15
5 THEN 1510
1506 IF A<Q65 OR A>Q90 THEN A=Q32
1507 IF A=Q32 OR A=Q126 THEN ? "ß":CLO
SE #Q1:G$="":GOTO 2005
1508 ? CHR$(A);:G$(LEN(G$)+Q1)=CHR$(A)
:NEXT D
1509 IF D>12 THEN A=Q32:GOTO 1507
1510 IF G$="" THEN A=Q32:GOTO 1507
1511 IF G$=TEMP$ THEN CLOSE #Q1:G$="":
GOSUB 1950
1512 CLOSE #Q1:GOSUB 2000
1950 GRAPHICS Q2+16:POKE 712,144:POKE
710,28
1951 GOSUB BRK:POSITION Q4,Q3:? #Q6;"t
hat\s right":RA=RA+Q1:WA=WA+Q1
1952 POSITION Q5,Q9:? #Q6;RA;" OUT OF
";WA
1953 POSITION Q9-LEN(NAME$)/Q2,Q6:? #6
;NAME$;"!!"
1954 SOUND Q0,83,Q10,Q10:FOR D=Q1 TO Q
20:NEXT D:SOUND Q0,60,Q10,Q10:FOR D=Q1
 TO Q20:NEXT D
1955 SOUND Q0,47,Q10,Q10:FOR D=Q1 TO Q
20:NEXT D:SOUND Q0,Q40,Q10,Q10:FOR D=Q
1 TO Q40:NEXT D
1957 SOUND Q0,47,Q10,Q10:FOR D=Q1 TO Q
20:NEXT D:SOUND Q0,Q40,Q10,Q10:FOR D=Q
1 TO Q40:NEXT D:SOUND Q0,Q0,Q0,Q0
1960 POSITION Q4,Q4:? #Q6;T$:POSITION
Q7,Q7:? #Q6;T$:FOR D=Q1 TO 35:NEXT D:C
OUNT=COUNT+Q1:IF COUNT=Q7 THEN 4060
1962 X0=INT(RND(0)*15):GOTO PICK
2000 POKE 710,Q0:? "ß":? "    Well, ";
NAME$;". . . Try again!":WA=WA+Q1
2001 SOUND Q0,84,Q10,14:FOR D=Q1 TO Q1
5:NEXT D:SOUND Q0,101,Q10,14:FOR D=Q1
TO Q15:NEXT D:SOUND Q0,Q0,Q0,Q0
2003 FOR D=Q1 TO 150:NEXT D
2005 FOR D=Q1 TO Q32:NEXT D:G$="":X0=P
EEK(712):POKE 710,X0:? "ß":GOTO Q90
```

```
4000 FOR J=Q0 TO PN:POOL(J)=J:NEXT J:F
OR J=PN TO Q0 STEP -1:K=INT(RND(Q0)*(J
+Q1)):PICTURE=POOL(J):POOL(J)=POOL(K)
4010 POOL(K)=PICTURE:NEXT J:PICTURE=(P
ICTURE*100)+99
4015 IF PICTURE=HOU THEN GOTO PICK
4016 IF PICTURE=STA THEN GOTO PICK
4017 IF PICTURE=BOX THEN GOTO PICK
4018 IF PICTURE=KIT THEN GOTO PICK
4019 IF PICTURE=BOO THEN GOTO PICK
4020 IF PICTURE=TV THEN GOTO PICK
4021 IF PICTURE=TRU THEN GOTO PICK
4022 IF PICTURE=FOOT THEN GOTO PICK
4023 IF PICTURE=FAC THEN GOTO PICK
4024 IF PICTURE=CON THEN GOTO PICK
4025 IF PICTURE=LOL THEN GOTO PICK
4030 GOTO PICTURE
4060 IF WA=Q7 THEN REWARD$=" PERFECT!!
"
4062 IF WA=Q8 THEN REWARD$=" GREAT!! "
4064 IF WA=Q9 THEN REWARD$=" GOOD!! "
4066 IF WA>Q9 THEN REWARD$=" PRETTY GO
OD!! "
4070 GRAPHICS 0:POKE 710,212
4071 ? :? :? :? ;REWARD$;:? :? :? ;NAM
E$;", you answered ";RA;" out of ";WA:
? "questions correctly!"
4074 GOSUB BRK:POKE 752,Q1:POSITION Q5
,Q10:? "Level    ";LV:POKE OFF,Q1:POK
E KEY,Q255:SPOT=4079
4076 ? :? :? "Would you like to try ag
ain? (Y/N) ";:OPEN #Q1,Q4,Q0,"K:":GOS
UB BRK
4077 GET #Q1,A:IF A=89 THEN CLOSE #Q1:
RUN
4079 POKE OFF,Q0:POKE 16,192:POKE 5377
4,247:GRAPHICS 0:CLR :POP :END
5000 ? ,"      HOUSE ":RETURN
5010 ? ,"       STAR ":RETURN
5020 ? ,"       BOX ":RETURN
5030 ? ,"       KITE ":RETURN
5040 ? ,"       CONE ":RETURN
5050 ? ,"       BOOK ":RETURN
5060 ? ,"       TV  ":RETURN
5070 ? ,"     LOLLIPOP ":RETURN
5080 ? ,"      TRUCK ":RETURN
5090 ? ,"    FOOTBALL ":RETURN
5100 ? ,"       FACE ":RETURN
6000 ? "  HOUSE          CAR          B
OAT ":RETURN
6010 ? "   TRUCK          GIRL         S
TAR ":RETURN
6020 ? "   TABLE          BOX          T
OP ":RETURN
6030 ? "   SHOE          TAIL          KIT
E ":RETURN
6040 ? "   CONE          HELLO         TR
EE ":RETURN
6050 ? "    BOOK         RECORD         G
LASS ":RETURN
6060 ? "   OVEN           TV           CHA
IR ":RETURN
6070 ? "   DESK          PAPER         LOLL
IPOP ":RETURN
6080 ? "   TRUCK          STAR         B
ALL ":RETURN
6090 ? "    BAT         FOOTBALL        H
AT ":RETURN
6100 ? "  GAME           LID          F
ACE ":RETURN
7000 ? "HORSE    HUNCH    HOUSE
HOW ":RETURN
7010 ? "  STOP    STAY    STAR    S
TART ":RETURN
7020 ? "BOX      BOY      FOX      BL
OCKS ":RETURN
7030 ? "  KITTEN    KIT    KITE    K
ITTY ":RETURN
7040 ? "  CANE    CART    BONE    C
ONE ":RETURN
7050 ? "TOOK      BOO      BOOK    B
OMB ":RETURN
7060 ? "  T.B.    TV    TENT    TEE
VEE ":RETURN
7070 ? "LULU    LULL    LOLLIPOP    L
ARD ":RETURN
7080 ? "CAR     TRUCK     TRAIN     T
ROLL ":RETURN
7090 ? "FEEDBAG    FOOT    FOOTBALL
  FOOD ":RETURN
7100 ? "FOOT     FACE     FLASH
FADE ":RETURN
8000 ? :RETURN
13000 POKE 16,64:POKE 53774,64:POKE 70
2,64:POKE 694,Q0:RETURN
13500 A=Q32:GOTO SPOT
14000 GRAPHICS Q6:GOSUB BRK:COLOR Q1:P
OKE OFF,Q1:POKE 710,(X0*16)+Q4:POKE 71
2,PEEK(710):RETURN
15000 Q1=1:Q0=Q1-Q1:Q2=Q1+Q1:Q3=Q2+Q1:
Q4=Q3+Q1:Q5=Q4+Q1:Q6=Q5+Q1:Q7=Q5+Q2:Q8
=Q7+Q1:Q9=Q8+Q1:Q10=Q9+Q1:Q15=Q10+Q5
15005 Q20=Q10+Q10:Q29=Q20+Q9:Q32=Q20+Q
10+Q2:Q40=Q20+Q20:Q49=Q40+Q9:Q52=Q40+Q
10+Q2
15010 Q65=65:Q90=90:Q126=126:Q155=155:
Q255=255:CUE=1500:CLUE=5000:PICK=4000:
BRK=13000:OFF=752:KEY=764:KOLOR=14000
15020 QTRAP=13500:PN=Q10:DIM NAME$(12)
,REWARD$(15),TEMP$(Q10),G$(15),T$(12),
POOL(PN):T$="                "
15030 RETURN
16000 GRAPHICS Q8:GOSUB BRK:POKE 710,9
8
16010 ? "  This program will show you
some     pictures and picture words. T
here are four difficulty levels."
16020 ? :? :? "LEVEL 1  This level s
imply asks you to type the given pictu
re word."
16030 ? :? :? "LEVEL 2  This level g
ives you some  words to choose from to
 find the      picture word."
16040 ? :? :? "LEVEL 3  This level g
ives you some  very similar words to c
hoose from."
16050 ? :? :? "LEVEL 4  This level a
sks you to     spell the picture word
without any    clues.":RETURN
```

•

CHECKSUM DATA

(See page 23)

```
0 DATA 890,841,269,928,121,895,855,674
,815,647,244,785,377,17,670,9028
27 DATA 960,695,908,269,165,174,742,74
2,745,164,671,51,60,790,775,7911
65 DATA 776,390,59,480,567,619,998,630
,40,526,680,692,64,689,662,7872
115 DATA 973,695,694,418,35,422,363,32
8,426,51,355,41,845,434,591,6671
601 DATA 620,118,205,175,656,438,541,6
41,387,222,969,313,730,95,404,6514
806 DATA 270,577,591,459,217,446,331,8
29,27,912,651,885,375,227,736,7533
940 DATA 794,959,744,393,54,751,661,30
4,64,895,166,747,176,984,898,8590
1028 DATA 724,995,111,727,295,890,730,
681,641,527,516,529,393,679,33,8471
1110 DATA 725,344,209,215,30,994,325,9
0,254,761,392,236,356,794,709,6434
1954 DATA 157,713,801,194,129,631,243,
378,31,620,256,854,854,846,861,7568
4019 DATA 832,576,886,808,795,824,849,
425,805,740,626,210,726,820,588,10510
4076 DATA 337,352,556,241,48,480,770,7
67,3,136,466,256,509,719,937,6577
6010 DATA 686,903,898,259,119,501,927,
612,667,557,685,749,248,440,996,9247
7050 DATA 880,838,402,739,427,161,94,6
40,871,239,338,743,583,526,61,7542
16000 DATA 330,390,104,853,982,44,2703
```

•

MORE DISK DRIVE FOR YOUR MONEY .....

In fact, with the ASTRA 1620, you get two superb Disk Drives for the price of one. The ASTRA 1620 is Single or Double Density (software selectable) and completely compatible with ATARI DOS or OSA+ DOS. When used as Double Density, the ASTRA 1620 has the same capacity as Four ATARI 810® Disk Drives.

DOUBLE OR SINGLE DENSITY

The ASTRA 1620 can be either single or double density, depending on the software selected. One drive can be configured for single density and the other drive for double density, or any combination desired. The ASTRA 1620 is compatible with virtually any software available for ATARI® Disk Drives. The ASTRA 1620 is smooth, quiet and fast. In Single Density mode, the ASTRA 1620 stores 88K bytes of programs or files. In Double Density, the ASTRA 1620 stores 176K bytes, simply twice as much.

TWO DRIVES ... Yes, two superb disk drives in the same size enclosure normally used for one drive. The ASTRA 1620 measures 7⅞" wide x 11⅞" deep x 5⅞" high.

Two drives will open a new dimension of computing for you. The program disk can be in one drive and the data disk can be in the other. This will eliminate time consuming disk changes. Backing up disks and copying files will never be easier. Just follow the instructions on the screen and walk away. The job will be completed within minutes. We have simplified copying from single to double density. With two drives, it's just as easy as copying in one mode. No disk switching!

Two double density drives give you the power that much larger and more expensive computers have without giving up any of the features available on the ATARI® Home Computer.

EASY TO USE ... The ASTRA 1620 comes complete with everything you need. Just plug it in, chain it up, and turn it on.

The ASTRA 1620 comes with OSA+ DOS (The best disk operating system available for the ATARI® computer!). The OSA+ DOS is completely compatible with all existing ATARI DOS files. Because the OSA+ user manual is very complete and technical, we include our own simplified user manual. Between the two furnished manuals, you have the information necessary to perform any task required of your disk drive.

The ASTRA 1620 also contains a data cord, power transformer, and operator manual.

ASTRA SYSTEMS

5230 Clark Avenue, Suite 19
Lakewood, California 90712

Phone
(213) 804-1475

CIRCLE #104 ON READER SERVICE CARD.

AlterDOS

16K Disk

by Gordon L. Banks

PROBLEM: Suppose that you, as many of us do, have several AUTORUN.SYS files that are twelve sectors long. Since the filename and the length are identical, how to you remember what each one does? Usually you have to boot that disk to find out. This is too time consuming.

SOLUTION: Modify your DOS to recognize filenames other than just AUTORUN.SYS as the autorun file.

HOW: When DOS is booted and loaded into RAM, memory locations 5903 through 5914 contain the name recognized by DOS as the autorun file. All we have to do is POKE in our own preference, and then SAVE the modified version of DOS back onto the disk. The following program, **AlterDOS**, does just that. But what new name do we use? One idea with merit is to alter DOS to recognize any eight-character filename with the .ARS extender. With the asterisk (*) wildcard replacing the eight-character filename, you can use those eight characters to give your autorun files meaningful names, such as RENUMBER, or SCRNDUMP, or whatever you like. Any name would work, as long as it has the .ARS (AutoRunSys) extender. Then you could also keep additional autorun files on the same disk, but in an inactive status, by changing the extender to something like .ARF (AutoRunFile). Your DOS-copying functions would also copy these more readily than files with the .SYS extender.

However (and you just *knew* there would be a "however," didn't you?), there is a serious drawback to this plan. Files still named AUTORUN.SYS won't work until renamed. This means renaming all current and future AUTORUN.SYS files before you use them. Swapping disks with friends will eventually lead to your modified DOS being in the hands of someone else. Think of the problems they will experience when their AUTORUN.SYS files won't work. Also, please believe me, there will come a time when you will forget and try to boot up a disk with an AUTORUN.SYS file, and when it doesn't work, you do a lot of head scratching.

My recommendation is to change your autorun identifier to A*.SYS. This method still allows you to use seven characters with which to define your autorun files with meaningful names. For instance, isn't ARENUMBER.SYS or ASCRNDMP.SYS really an improvement? This way, if a friend winds up with your modified version of DOS, it won't matter, and an autorun file named AUTORUN.SYS (yecch!) will still be recognized by DOS, just as usual.

So, consider these two possibilities — along with your own ideas. Maybe you'll come up with something better and share it with the rest of us. Just insert your preference into Line 160 where I have "A*.SYS". □

```
10 REM .........ALTERDOS..........
20 REM ..by Gordon L Banks.........
30 REM ............................
40 GRAPHICS 0:DIM A$(40):POKE 752,1
50 READ A$:IF A$="*" THEN 160
60 ? A$:GOTO 50
70 DATA              ALTERDOS
       ,
80 DATA This program will alter your D
OS to
90 DATA allow more descriptive names f
or your
100 DATA AUTORUN.SYS files. You will h
ereafter,be able to use names that inc
lude the
110 DATA letters of your choice in lie
u of the,'x's shown in this example.
```

```
120 DATA ,          D:Axxxxxxx.SYS,,(No
te that a file named AUTORUN.SYS, will
 still function as normal.),
130 DATA Now names such as ARENUMBR.SY
S or,ASCRNDMP.SYS (for RENUMBER or SCR
EEN-,DUMP) may be used.
140 DATA ,Just remember to start the n
ame with,an A and end with .SYS.,
150 DATA If you are ready press RETURN
.,,*
160 INPUT A$:RESTORE 180:A$="A*.SYS"
170 IF LEN(A$)>12 THEN ? :? "■ NO MORE
 THAN 12 CHARACTERS ALLOWED.":GOTO 230
180 FOR I=1 TO LEN(A$):POKE 5902+I,ASC
(A$(I,I)):NEXT I
190 FOR I=5903+LEN(A$) TO 5914:POKE I,
155:NEXT I
200 ? :? " Now writing new DOS.SYS fil
e."
210 REM XIO 36,#1,0,0,"D:filename.ext"
 unlocks file if locked.
220 XIO 36,#1,0,0,"D:DOS.SYS":CLOSE #1
:OPEN #1,8,0,"D:DOS.SYS":GRAPHICS 0:?
"I'M DONE,":? "AND YOU'RE";
230 POKE 752,0:END
```

●

CHECKSUM DATA

(See page 23)

```
10 DATA 225,395,768,444,303,958,831,64
2,270,555,878,432,653,294,678,8326
160 DATA 820,688,583,835,346,643,778,6
51,5344
```

●

Auto Line Numbering +

16K Cassette or Disk

by Sam Wiley

If you're like me, you love to write programs but hate to even *think* about the mental and physical work involved in keeping track of line numbers as you program. After all, we geniuses need to think about program logic and not the next line increment!

This utility will do four things. First, it will put line numbers on the screen. Second, it will check the line that you typed in for an error and BEEP you by ringing the bell. This is also known as printing CHR$(253) or hitting CTRL-2. Third, it displays, near the top of the screen, how many variables you have left and, also, how much memory you have left. Fourth, it disables the BREAK key and the clear screen keys. There is also full-screen editing of the lines that are on the screen. You can change the starting line number, or the increment, by pressing CTRL-3 and typing GOTO AUTO. After hitting CTRL-3, you can LIST your program, edit it or add statements. The utility will go back to the next line that it was working on, before CTRL-3 was pressed, by typing GOTO NEXT.

Although most of it is in BASIC, it uses the "forced read mode" to read the entire screen every time RETURN is pressed. This is what allows for full screen editing. There is a short machine language subroutine to check the entire BASIC program for an error and find out how many variables have been defined. Both of these functions work very fast, so there is hardly any wait while you are entering your program. The program that you are typing in can be separated from the auto-numbering utility by LISTing it to the disk or cassette with the line numbers 0-31999. Here is an example: LIST "D:YOURPROG.EXT",0,31999. This will only LIST your program to the disk. For a cassette-based system, use LIST "C:",0,31999. Lines 32045 and 32055 contain the REM equivalent of these statements. I suggest you type in whichever one applies, and then you can delete the line number and the REM. For disk users, fill in the name that you want to call your program on Line 32045. Press RETURN, and it will be properly LISTed to disk. Cassette users can use Line 32055. By using the LIST command, you will have to use the ENTER command to get your program back into the computer. I suggest you use the SAVE command to save this utility and the LIST command to save your program. This way you can always merge the two by loading the utility first and then using ENTER to load your program.

The machine language subroutine is placed in a string (ML$) to allow access to page 6. The BREAK key is disabled, because it was put too close to the RETURN and editing keys on the 800. Instead you use CTRL-3 to interrupt the auto-numbering process. This will also restore the BREAK key for normal editing.

Type in the program and use one of **ANALOG**'s error-checking programs, **C:CHECK** or **D:CHECK**. I just can't say enough about this idea for typing in programs from printed media. Until they came along, I always thought that the magazine was printing the wrong code. No way. I learned what a really lousy typist I was. At any rate, after the CHECKSUM DATA checks out, type RUN. The screen will go blank for a few seconds, while the utility loads the machine language subroutine into the string. The first thing that prints is "STARTING LINE NUMBER ?" Answer with the line number you want to begin auto line numbering with. If you don't answer with a number, the utility will repeat the question. The second thing you are asked is "INCREMENTS DESIRED?" Answer with the number of lines that are to be between each line number. A good number is 10. This way you can insert 9 lines between each line that you type in, if you find out later that something new should be added.

Next, the first line will appear, and the utility will wait for you to enter a statement. All of the keys are accepted, with the exception of CTRL-CLEAR and SHIFT-CLEAR. These keys are useless during screen editing. If the ESC key is pressed prior to CTRL-CLEAR or SHIFT-CLEAR, it will accept the key(s). When RETURN is pressed, the screen blanks for a second. If there was no error in the line, it will click the console speaker to alert you that it is ready to accept the next line. This is so you don't have to look away from whatever you are typing in. At the top of the screen will be the utility commands. They are: 1. CTRL-3 — Use instead of BREAK key; 2. G.NEXT — GOTO next line number of a predefined increment; and 3. G.AUTO — Set up new line numbers and increments.

Under this is displayed how many variables are left and how much memory is left. The memory is displayed as "BYTES LEFT=". The last two lines are displayed, along with the next line. If there was an error in the line you typed in, the console bell (CTRL-2) will ring, and the only thing on the screen will be the error line and statement following it. The cursor is positioned at the beginning of the line number. Be sure to remove the ERROR- and also the cursor that shows where the error is, to avoid getting another error. When it is corrected, press RETURN, and the utility will go to the next line number.

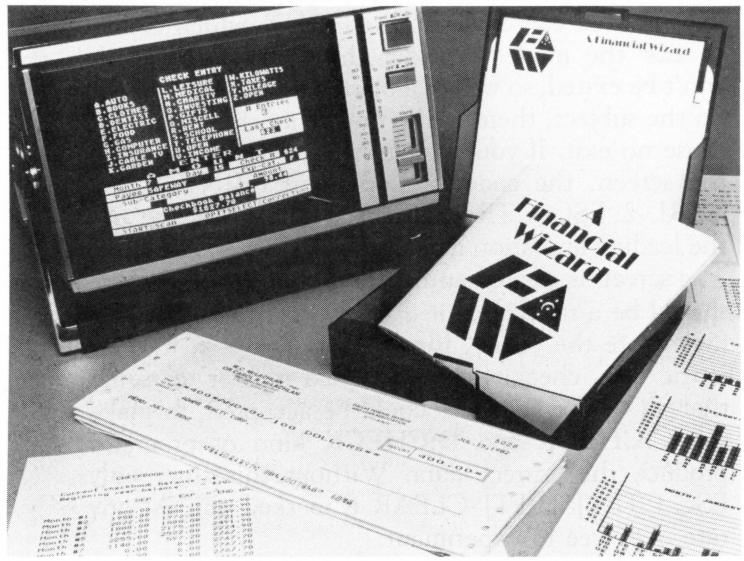
You may also use the utility to delete lines of code. Type GOTO AUTO and give the starting line number and increments of the lines that you want deleted. Then just hit RETURN when the line numbers appear. I also use it to type in programs from magazines and books. Sometimes the line numbers are not in even numerical increments (like in this program). When this is the case, you have to use the GOTO AUTO to keep changing the starting line number and the increments. For myself, it's most useful for program development from scratch.

You may wonder why the utility only LISTs two of the previous program lines. In studying the listing, you will see that Line 32113 can be modified to LIST three or even four previous lines. For instance, after the TRAP statement, a LIST LINE-INC*3 will LIST the last three lines. The reason for the two lines is due to the Atari being in thue "forced read mode. "Even though the CONT is at Line 19, if there are three full lines on the screen (a logical line can be four physical lines), and the fourth line contains an error, the error message could cover up the CONT, and the computer never regains consciousness until SYSTEM RESET is pressed. In the "forced read mode," if a line is encountered with an error, the line is immediately displayed again with the ERROR- message.

(continued on page 62)

If you are serious about personal finance...

- **Budget Forecast** – 26 expense categories
- **Check Entry** – easy data entry – scan & modify 26 major & 36 sub-categories – information block
- **Check Search** – single or multiple parameters – (up to seven) to search entries
- **Tabulations** – detailed expense vs. budget comparisons by month, year-to-date, category
- **Bar Graphs** – screen displays in graph form expenses vs. budget – by month or category – printing with graphic capable printers
- **Check Reconciliation** – fast clearing of resident checks & deposits, complete summary report
- **Checkwriter** – prints your custom checks
- **Printouts** – most popular printers
- **Multi-Colored Graphics • Audio Enhancements**
- **7 Utility Programs • User-Friendly Operation**
- **Easy To Use Instruction Manual • Audit Report**
- **Handsome Tinted Plastic Storage Case**



COMPUTE!
The Leading Magazine Of Home, Educational, And Recreational Computing

In a feature editorial.

"If you want to use a finance system, but don't want to spend several days trying to learn how to use one, then A Financial Wizard by Computari may be just what you need."

"The illustrated manual that comes with this program is clear, direct, and very thorough."

"It appears that this finance system was designed to achieve the best and most comfortable working relationship between the user and the program."

"The check entry routine is the most attractive feature of this finance system. Data prompts are very clear and the category item names are displayed at all times during data entry for your convenience."

"The file search capabilities of this program are superior. You are offered seven ways to look up the checks."

"The system is disk intensive. All data is saved automatically and immediately following all routines that either enter data or modify it."

"Scanning your entries is made possible by pressing START. You can see records very quickly this way."

"This is an excellent finance system – entertaining, accurate, and fun to use."

ANALOG COMPUTING
THE MAGAZINE FOR ATARI COMPUTER OWNERS

Analog Magazine in a comprehensive study of personal finance systems for Atari* computers.

"A Financial Wizard from Computari is by far the best of these programs and will be the standard of comparison for the others."

"The check entry mode is easy to use..."

"The way a Financial Wizard handles your tabulations is excellent. You can chart your actual expenses vs. your budget by month, by category or year to date."

"...where it really outshines the rest is in the check reconciliation."

"In effect it gives you your bank statement on the screen, a complete list by month of all your checks and deposits."

"A Financial Wizard has one disk that does everything..."

"Graphics, while really not a factor in the quality of programs of this type, do make your budgeting chores a little more pleasant. Again A Financial Wizard comes out on top."

"Everything about this program is excellent..."

Antic
The ATARI Resource

In a Report from Antic.

"Like most Atarians, I am captivated by the graphic, color and sound capabilities of my machine. Nothing quite discourages me more than to boot up an applications program (personal, business, etc.) and to be presented with the standard graphic 'o' white characters on a blue screen.

Of course the usefulness and effectiveness of a program is of primary importance. However, enhancing the dullest of applications programs with some of Atari's charms, is a great asset. A Financial Wizard, a personal finance program by Computari's Bill McLachlan, is an excellent example of an applications program that integrates many of the Atari's features into a well conceived and executed program."

"The use of color and sound in the data input prompts and error checking routines are so well done that it's quite simple to boot up the disk, follow along with the very clear documentation, and be 'up and running' in short order."

"I give A Financial Wizard high marks in ease of use, documentation and performance. If a disk-based home finance package is in your future, The Wizard should get serious consideration."

Computari's
A Financial Wizard 1.5
The logical choice.

The system is designed for Atari computers having a minimum of 32K and operating from a disk drive. The cost is only $59.95 plus $3 for handling/postage.

If your dealer does not have **A Financial Wizard** . . . Telephone orders are accepted on Mastercharge or Visa credit cards. Mail order must be accompanied by check or money-order or credit card #.

Dealer inquiries invited.

*trademark of Atari Inc.

OK residents ad 4% Tax.

Exclusively thru **ON LINE** COMPUTER CENTERS of OKC ™ (405)751-7925

10944 North May, Oklahoma City, OK 73120

☐ CHECK ENCLOSED ☐ VISA ☐ MASTERCHARGE

NAME (Print)_____

Address _____

City _____

State _____ Zip _____

Card # _____ Exp._____

Signature_____

This is what can cover up the CONT, which is used to exit the mode. Without the CONT, the mode can't be exited, so we get a blank screen. While we're on the subject, there is also one other thing that can cause no exit. If you want to ring the bell and clear the screen, the code is: [Linenum] PRINT "[ESC CTRL-2 ESC CTRL-CLEAR]". If you leave out the leading quotation mark, an error line is generated. The screen is cleared and ZAP! No CONT. This one should be a rarity, but it did happen to me. When I first wrote the utility, Line 32081 wasn't in it. This is the ESC checking line. I added it later to insure that all of the keycodes could be accepted, to make the **C:CHECK** and **D:CHECK** kind of programs produce the correct data. Without this line, only ESC CTRL[SHIFT]-CLEAR is masked out. At any rate, feel free to experiment.

The utility consumes 27 of the 128 variables that you can possibly have. I did it this way to conserve RAM. If you need more than 101 variables, you can substitute the variables on Line 32061 to numbers and change them in the program.

If you type RUN after you have keyed in some or all of your program, you will need to type GOTO 32000 to restore the auto line numbering utility. □

Program description.

Lines 32060-32061 — Initialize variables and set screen color to blue when ANTIC is turned off. (Whatever color is in 712 when you do a POKE 559,0 is the color the whole screen turns.) Use READ statement to assign variables to statement commands. (NEXT, ON, CLR and GETKEY would not normally be allowed.) Use READ statement to assign numbers to variables that are frequently used. Assigning the value of 1 to C1 and using it instead of a 1 saves memory (but uses a variable) every time it is used instead of the number 1.

Lines 32064-32067 — Load machine language subroutine that checks BASIC program for an error and the number of variables used.

Lines 32068-32075 — Open keyboard for input, disable BREAK key, clear screen, make speaker click, and get starting line number and increments. Print first line number.

Lines 32080-32085 — GETKEY routine. Reads keyboard and prints to screen until a RETURN is encountered. Masks out CTRL-CLEAR, SHIFT-CLEAR. Checks for ESC key.

(Continued on page 64.)

RAM Has More In Store

We're The Experienced Generation Of ATARI Experts!

- Software—Over 1000 Titles!
- Hardware
- Educational & Business Consultations
- Authorized Service
- Instruction
- 24-Hour Bulletin Board at (617) 371-1855
- Shipments Everywhere

- ATARI 400 Replacement Full-Stroke Keyboard: **$69.95**

Special Savings

- Entire Line Of Electronic Arts Software Including:
  Music Construction Set
  Seven Cities of Gold
  One On One (Doctor J and Larry Bird)
  Word Flyer
- Sublogic Flight Simulator

Begin word processing with your ATARI easily and inexpensively with one of these specially tailored packages. You'll increase your personal productivity and also save money!

For ATARI owners without a disk drive:

- Track AT-D2 w/Printer Interface
- Interface Cable
- ATARIWRITER
- Printer Driver From APX
- Mannesmann Tally Spirit-80 Printer

$799.00 (save over $100.00)

For those who own a disk drive already:

- Ape Face Printer Interface
- ATARIWRITER
- Printer Driver From APX
- Mannesmann Tally Spirit-80 Printer

$499.00 (save over $60.00)

New England's #1 ATARI Source

RAM COMPUTER CENTER

427 Great Road, Acton, MA 01720 (617) 263-0418

CIRCLE #133 ON READER SERVICE CARD.

Lines 32100-32111 — Turn ANTIC off for speed. Set up screen for "forced read mode," and read. Stop "forced read" and exit to machine language subroutine, to check BASIC for error and variables. If no error occurred, increment line by the value in the variable INC.

Lines 32112-32114 — Clear screen. Print utility commands. Print number of variables and memory left. LIST last two lines and the next line number. Turn ANTIC back on and make the console speaker click. Return control to user for input of statements.

Line 32115 — This is where the utility goes when you type GOTO NEXT. This insures that the keyboard is closed and reopened to avoid a 129 or 133 error. Turn ANTIC off and branch back to LIST the last two lines and line number.

Line 32116 — Keyboard and BREAK disable subroutine.

Line 32120 — If we can't find a previous line number to LIST, we just display the current line number. Make sure ANTIC is on, and return to user for input.

Lines 32520-32525 — If there was an error found in the machine language subroutine, we perform this routine. Find out the address of the line number where the error is. Get the line number. Place it in the variable ERRLINE. Alert user by ringing bell. Clear screen. LIST the line with the error. Turn ANTIC on. Position cursor over line number. Return to user for input.

Line 32600 — DATA for the variables at Line 32061.

Lines 32700-32704 — DATA for the machine language subroutine.

Line 32710 — This is where the utility goes when CTRL-3 is pressed. Restore BREAK key and END.

Basic listing.

```
32000 REM ****************************
32001 REM *  AUTO LINE NUMBERING     *
32002 REM *  BY SAM WILEY SR.        *
32003 REM *  ANALOG COMPUTING        *
32004 REM ****************************
32005 REM
32020 REM 27 VARIABLES
32025 REM 1760 BYTES (REMS DELETED)
32030 REM TO SEPARATE AUTO UTILITY
32040 REM L."D1:PROGNAME.EXT",8,31999
32055 REM L."C:",8,31999
32056 REM
32057 REM TO GET PROGRAM BACK
32058 REM E."D1:PROGNAME.EXT"
32059 REM E."C:"
32060 RESTORE 32600:TRAP 32068:POKE 71
2,148
32061 READ NEXT,ON,GETKEY,ANTIC,AUTO,S
PEAKER,CLR,C0,C1,C2,C4,C53774,C256,C16
,C842,C203,C12,C13
32064 DIM ML$(90):A=C1:POKE ANTIC,C0
32067 READ N:ML$(A,A)=CHR$(N):A=A+C1:G
OTO 32067
32068 GOSUB 32116:? CHR$(CLR):POKE SPE
AKER,C0
32070 POKE ANTIC,ON:TRAP 32070:? "STAR
TING LINE NUMBER";:INPUT LINE
32075 TRAP 32075:? "INCREMENTS DESIRED
  ";:INPUT INC:? CHR$(CLR):? :? :? LIN
E;" ";
32080 TRAP 32710:GET #C1,KEY:IF KEY=15
5 THEN 32100
32081 IF KEY=27 THEN ? CHR$(KEY);:GET
#C1,KEY:GOTO 32085
32082 IF KEY=CLR THEN GOTO GETKEY
32085 ? CHR$(KEY);:GOTO GETKEY
32100 POKE ANTIC,C0:POSITION C2,19:? "
CONT":POSITION C2,C0:POKE C842,C13:STO
P
32110 POKE C842,C12:X=USR(ADR(ML$)):IF
 PEEK(207)=C2 THEN 32520
32111 LINE=LINE+INC:? CHR$(CLR)
32112 POSITION C2,C0:? "       CTRL/3
 G.NEXT   G.AUTO":? "Variables left=";
128-PEEK(205);"   Bytes left=";FRE(C0)
32113 ? :TRAP 32120:LIST LINE-INC*C2:L
IST LINE-INC:? :? LINE;" ";
32114 POKE ANTIC,ON:POKE SPEAKER,C0:GO
TO GETKEY
32115 GOSUB 32116:? CHR$(CLR):GOTO 321
12
32116 CLOSE #C1:OPEN #C1,C4,C0,"K":POK
E ANTIC,C0:POKE C53774,112:POKE C16,64
:RETURN
32120 ? LINE;" ";:POKE ANTIC,ON:GOTO G
ETKEY
32520 ERRADR=PEEK(C203)+PEEK(C203+C1)*
C256:ERRLINE=PEEK(ERRADR)+PEEK(ERRADR+
C1)*C256
```

```
32525 ? CHR$(253):? CHR$(CLR):LIST ERR
LINE:POKE ANTIC,ON:POSITION C2,C1:? :P
OKE 764,255:GOTO GETKEY
32600 DATA 32115,34,32080,559,32060,53
279,125,0,1,2,4,53774,256,16,842,203,1
2,13
32700 DATA 104,165,136,133,203,165,137
,133,204,160,1,177,203,48
32701 DATA 31,200,177,203,133,209,160,
4,177,203,201,55,240,13,24,165,209,101
,203,133
32702 DATA 203,144,228,230,204,208,224
,169,2,133,207,96,169,1,133,207,165,13
4,133,203
32703 DATA 165,135,133,204,169,0,133,2
05,165,203,197,136,208,7,165,204,197,1
37,208,1
32704 DATA 96,230,205,24,169,8,101,203
,133,203,144,232,230,204,208,228
32710 POKE C16,192:POKE C53774,247:? :
? :END
```

●

CHECKSUM DATA

(See page 23)

```
32000 DATA 817,853,381,628,829,564,188
,613,65,889,900,577,885,821,234,9244
32060 DATA 489,905,547,686,813,199,265
,153,32,582,86,942,593,267,511,7070
32113 DATA 83,424,117,529,817,770,866,
226,606,578,931,723,409,419,7498
```

●

Assembly language listing.

```
0100 ;    CODE IN DATA STATEMENTS
0110 ;      AT LINES 32700-32704
0120 ;
0130 ;   SEARCH THE BASIC PROGRAM
0140 ;   FOR ERRORS AND VARIABLES
0150 ;
0160 ;          SET UP EQUATES
0170 ;
0180 VARTAB=134         ;Var valu table
0190 NUMVAR=205         ;For BASIC
0200 STMTAB=136         ;BASIC's start
0210 NEXLINE=209        ;Storage
0220 ERRFLG=207         ;For BASIC
0230 POINTER=203        ;BAS zero page
0240 ERROR=55           ;BOO-BOO
0250 ;
0260 ;      POINT TO STATEMENTS
0270 ;
0280  *=1664            ;ASM use only
0290  PLA               ;Don't need
0300  LDA STMTAB        ;Set up zero
0310  STA POINTER       ; page pointers
0320  LDA STMTAB+1      ; for BASIC
0330  STA POINTER+1     ; search.
0340 ;
0350 ;          LAST LINE?
0360 ;
0370 NEXT LDY #1        ;End loop if we
0380  LDA (POINTER),Y   ; reached the
0390  BMI RETURN        ; last line.
0400 ;
0410 ;     GET NEXTLINE POINTER
0420 ;
0430 GETNEXT INY        ;Point to next
0440  LDA (POINTER),Y   ; NEXTLINE byte
0450  STA NEXLINE       ; and keep it.
0460 ;
0470 ;        CHECK FOR ERROR
0480 ;
0490  LDY #4            ;COMMAND offset
0500  LDA (POINTER),Y   ;Point to it
0510  CMP #ERROR        ;Error?
0520  BEQ RETURN2       ;Yes-Return a 2
0530 ;
0540 ;        UPDATE POINTER
0550 ;
0560  CLC               ;No error so
0570  LDA NEXLINE       ; let's get the
0580  ADC POINTER       ; next line's
0590  STA POINTER       ; address and
0600  BCC NEXT          ; go back to
0610  INC POINTER+1     ; check some
0620  BNE NEXT          ; more.
0630 ;
0640 ;       WE FOUND AN ERROR
0650 ;
0660 RETURN2 LDA #2     ;Tell BASIC we
0670  STA ERRFLG        ; goofed. (POKE
0680  RTS               ; 207,2)-- exit
0690 ;
0700 ;          NO ERROR
0710 ;
0720 RETURN LDA #1      ;Tell BASIC no
0730  STA ERRFLG        ; boo-boo's.
0740 ;
0750 ;      HOW MANY VARIABLES?
0760 ;
0770  LDA VARTAB        ;Set up zero
0780  STA POINTER       ; page pointers
0790  LDA VARTAB+1      ; for variable
0800  STA POINTER+1     ; search.
0810 ;
0820 ;          INITILIZE
0830 ;
0840  LDA #0            ;Put a zero in
0850  STA NUMVAR        ; our counter
0870 ;
0880 ;       LAST VARIABLE?
0890 ;
0895 NEXTVAR LDA POINTER; Last one
0910  CMP STMTAB        ; yet?
0920  BNE INCREM        ;No add 1 more
0930  LDA POINTER+1     ; How about
0940  CMP STMTAB+1      ; now?
0950  BNE INCREM        ;No add 1 more
0960  RTS               ;Now you may go
0970 ;
0980 ;     GET NEXTVAR POINTER
0990 ;
1000 INCREM INC NUMVAR;Add 1 more
1020  CLC               ;Get set to add
1030  LDA #8            ;Var # offset
1040  ADC POINTER       ;Let's update
1050  STA POINTER       ; our pointers
1060  BCC NEXTVAR       ; and continue
1070  INC POINTER+1     ; our search.
1080  BNE NEXTVAR       ;Uncond. branch
```

●

THE SEVEN CITIES OF GOLD
ELECTRONIC ARTS
2775 Campus Drive
San Mateo, CA 94403
(415) 571-7171
48K $39.95

by Arthur Leyenberger

There is no question that Electronic Arts is the premier game company for the Atari computer. They have been in existence roughly a year and have already produced a dozen titles. Many of these games have become classics. You know the ones I am talking about: **Pinball Construction Set**, **Archon**, **M.U.L.E.**, **Axis Assassin**, **Hard Hat Mack** and **Worms**.

M.U.L.E., which is an economic simulation taking place on a distant planet, was written by Dan and Bill Bunten of Ozark Softscape. Although difficult to believe, the Buntens have outdone themselves with their new game: **The Seven Cities of Gold**.

Seven Cities is a first-person simulation of sixteenth century Spanish conquistadors. After outfitting a ship and hiring a crew, you sail the oceans in search of new worlds. When land is sighted, you disembark with exploration parties in search of natives, treasures and the unknown. It is your decision to either trade with the natives or conquer them to obtain their valuable gold — which you would like to bring back to the homeland.

As the game begins, you obtain an audience at court, seeking gold to fund your expedition. Once you obtain the needed monies, you can stop at the pub for a refreshing brew while you contemplate your journey. Using the joystick you scroll out of the pub and pass by your home. It is here that you can assess your status, formulate plans and say goodbye to your spouse.

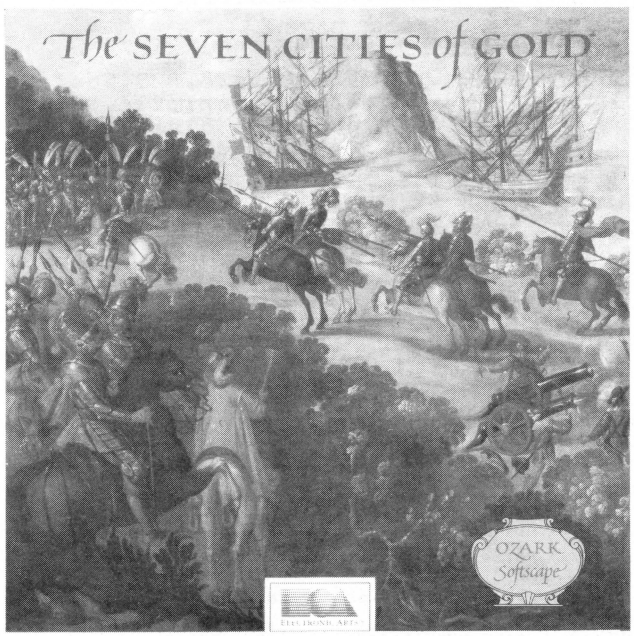
Next stop is the Outfitter, where you hire a crew, buy food and goods and purchase ships. Finally, you embark on your journey, and the court wishes you success.

As your voyage progresses, you must navigate by latitude and pay attention to the passage of time. Storms may be encountered, and lives may be lost due to sickness and storms. It is important to cross the ocean with the least expense of food and life. At any time you can view the ledger of your cargo in order to continually plan your journey.

Once land is sighted and you bring the ships into safe mooring, you must decide on how large an exploration party you want and what provisions you want to carry. Food is all important, but carrying too much will slow your journey. Goods are useful for trading with the natives. And enough men are needed — you may decide to establish forts and missions.

During your exploration of the local geography, you encounter rivers, lakes, plains and mountains. You can travel at various speeds, but travel and rough terrain will cost you additional food and may slow you down. Once you contact a native village, you have several choices. You may give them gifts, trade with them or conquer them. Trading is safer, but it is slower and requires many goods. Gift giving may not produce any immediate results, except show your good will and perhaps convince the natives to eventually tell you where a gold mine is.

Conquering is the easiest, but it will cost you lives and leave bitter memories. Also, the natives may communicate your hostile intent to other villages in the area. It is best to look for signs to determine the mood of the natives before you choose an approach that may have significant consequences.



The Seven Cities of Gold

After you have traded with or conquered several native villages, you will want to get back to your ship before your provisions run out. If you have not paid close attention to your route up till now, you may have trouble finding your ship. It is not uncommon for you and your exploration party to find yourselves lost in the jungle...and starvation is an unpleasant way to die.

If you do make it back to your ship — assuming they have not already sailed for home without you — you must transfer your supplies and booty back to the ship before you leave. Then you sail back across the ocean to your home port. Although your first stop may be the pub for a quick brew, you should visit your home to record your maps and review your journey. A trip to the court is in order to bear your treasures to the Queen. If you have done well,

you will bestowed with honors and maybe even a title. Now, if you can obtain additional funds, you may prepare for another journey.

Before you begin playing the game, you must create a map disk. You have your choice of using a map of the world as it was known in the 1500s or creating an entirely new world. Creating a new world takes about ten minutes and provides you with a much more challenging game. Rather than just creating random continents, the new world conforms to geological and cultural principles built into the program code.

There are several features that add to the playability of the game. Your current position may be SAVEd to the map disk at any time. You may then resume the game from where you left off. A new game may even be started with a map disk without disturbing the previously SAVEd game.

Seven Cities of Gold is an engrossing game. I have played it for hours at a time. Electronic Arts has certainly produced another high-quality game for the Atari computer. **Seven Cities** was written by Bill and Dan Bunten, Jim Rushing, Alan Watson and Roy Glover.

Now if you will excuse me, my crew tells me we are ready to set sail for the new world. Wish me luck. □

ATARI® DISK DRIVE HEADQUARTERS

No Penalty For Using Your Credit Cards!

COMPUTER PALACE

Formerly Royal Software

• DOUBLE DENSITY • PRINTER PORT

This is one of our best selling drives. It is reliable and quiet, and offers true double density storage. The AT-D2 model features a built-in parallel printer port with a 2K print buffer. The AT-1 is identical except it lacks the printer port and buffer. The AT-S1 is a slave drive for either the AT-D2 or AT-1 (no printer port or buffer).

TRAK AT-D2

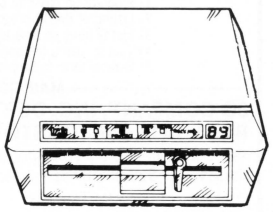
Our Price $398

PLUS FREIGHT — No Penalty For Charge Cards!

TRAK AT-1 ..... $349

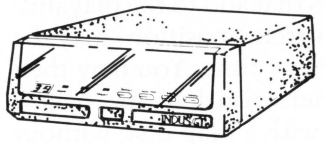
TRAK AT-S1 ..... $289

+ FRT.



• DOUBLE DENSITY • FREE SOFTWARE

We recommend and sell more of these drives than any other. It offers true double density, and is so quiet and dependable that you forget that you are using a disk drive. A Beautiful Jet Black finish with flip-up smoked Plexiglas dust cover and LED readout make it a visual delight. Add to this, the FREE SOFTWARE (Word Processing, Spread-Sheet and Data-Base) and a Full One Year Warranty and you have a winner. Includes DOS-XL.



INDUS GT

Our Price $398

PLUS FREIGHT — No Penalty For Charge Cards!

☆ **NO. 1 BEST SELLER!**

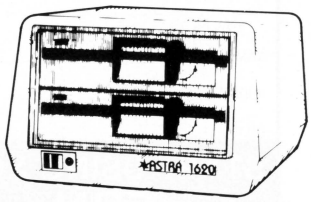
ASTRA 1620

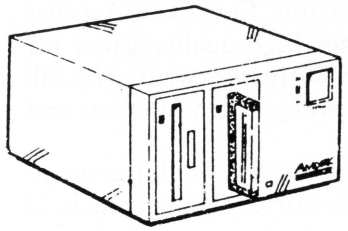
Our Price $529

PLUS FREIGHT — No Penalty For Charge Cards!

- **DOUBLE DENSITY**
- **DOUBLE DRIVES**

This drive packs a lot of storage into a small package which is only a little larger than the Atari 810 drive. It features two (2) double-density drives with the storage capacity of four (4) Atari 810's! It offers two (2) I/O ports to allow daisy-chaining of another drive or other peripherals, and the colors of the housing match the new XL series computers. All in all, this is a whole lot of disk storage for the money. Includes DOS-XL.





AMDEK MICRO DRIVE

- **DOUBLE DENSITY**
- **DOUBLE SIDED**
- **3" MICRO-FLOPPY**
- **PRINTER PORT**
- **DRIVE CONTROLLER**

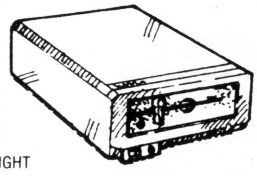
Now you can add the convenience and storage capacity of the new 3" micro-disk technology to your Atari. Each disk-cartridge can store 108K, and several software companies are releasing programs in this format (LJK, Continental, Broderbund, Penguin, for example). Also featured is a built-in parallel printer port and a disk drive controller which allows you to run any 5¼" drive including the double-density/double-sided RCP 100 and 200 shown below left. Includes DOS-XL.

AMDCI ..... Single Drive — No Penalty For Charge Cards! **$499**

AMDCII ..... Double Drive — No Penalty For Charge Cards! **$699**

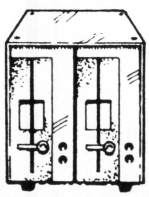
ATARI 1050

- **DUAL (1½) DENSITY**



Only $379 PLUS FREIGHT

RCP Add-On Drives



- **Double Density**
- **Double Sided**

For use with AMDEK or ATR-8000

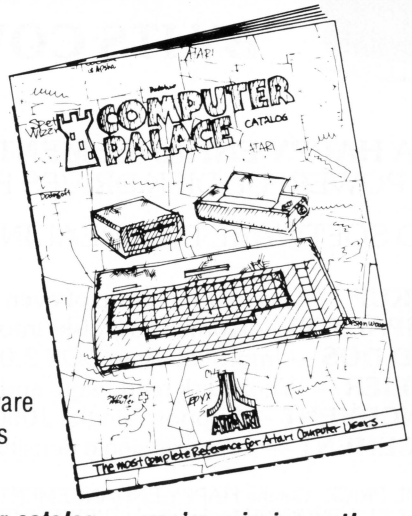
RCP 100 (1 Drive) **$379** PLUS FRT

RCP 200 (2 Drives) **$595** PLUS FRT

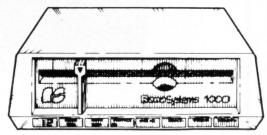
FREE CATALOG

with any order . . . or send $1 (refundable with first purchase).

This is the most comprehensive Atari reference catalog available! It contains over 3000 software & hardware listings with illustrations and descriptions!



If you don't have our catalog . . . you're missing out!



RANA 1000

- **Double Density**
- **One Year Warranty** (ATX Series)

$349 + FRT.

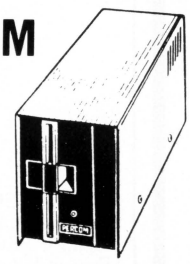
PERCOM

Model 88-S1PD

- **DOUBLE DENSITY**
- **PRINTER PORT**

$479 PLUS FRT

40-A1 Slave Drive $349



COMPUTER PALACE

OPEN M-F, 9-6 Sat. 10-4 (Pacific Time)

2160 W. 11th Avenue Eugene, Oregon 97402

MasterCard VISA AMERICAN EXPRESS

USE YOUR CREDIT CARD & CALL

Toll Free 1-800-452-8013

★ ORDERS ONLY, PLEASE ★

There's never a penalty for using your credit card!

For Information, Call (503) 683-5361

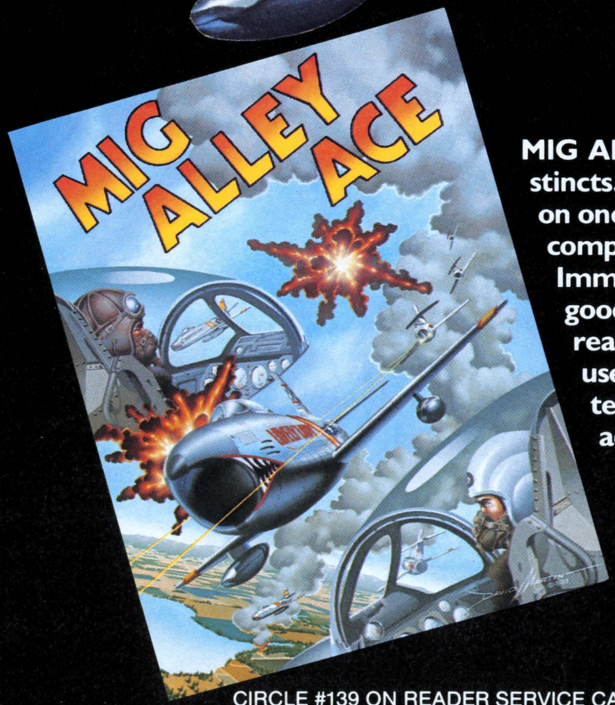
SHIPPING INFORMATION: Software — UPS ground ($2.90), UPS Air ($4.75) per order. Applies to software only. **Hardware, Foreign & APO**—Extra, based on actual weight of item(s). Please call (503) 683-5361 or write for information.

CIRCLE #138 ON READER SERVICE CARD.

ONE-ON-ONE!

EXPERIENCE THE THRILL OF REALISTIC 3-D HEAD-TO-HEAD AERIAL COMBAT!

MIG ALLEY ACE



MIG ALLEY ACE excites, stimulates, and challenges your fighter pilot instincts. It is the first simulation that allows you to challenge a friend to one on one combat and **SHOOT HIM DOWN!** Battle another human pilot, or computer controlled enemies. Roll inverted, execute a Split S, Loop, Immelmann, or any manuever you can, but don't let the enemy get a good shot at you! Outstanding action, superb 3-D graphics and deadly realism make **MIG ALLEY ACE** a must for all serious ATARI computer users. Available in disk or cassette for $34.95 at most computer centers, or order direct with MasterCard/Visa, C.O.D., or check. Please add $2.50 for postage & handling.

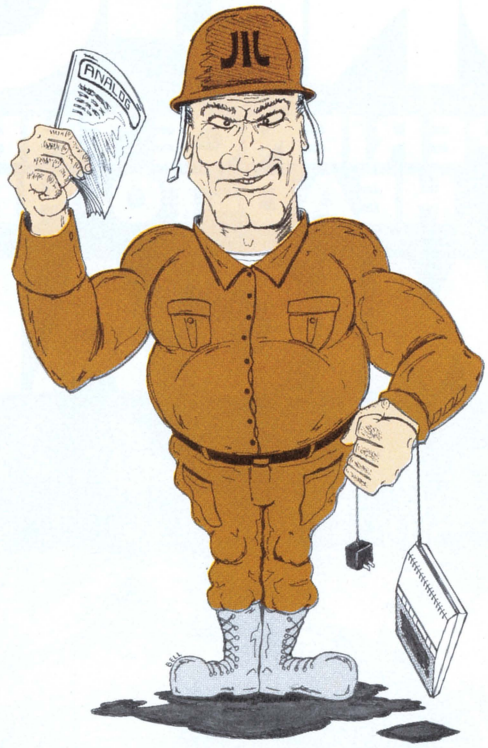
Look for MicroProse's other exciting real time simulators, NATO COMMANDER and SOLO FLIGHT.

EXPERIENCE THE
MicroProse Software
CHALLENGE!

10616 Beaver Dam Road
Hunt Valley, MD 21030
(301) 667-1151

CIRCLE #139 ON READER SERVICE CARD.

BASIC Training



by Tom Hudson

In this issue's **BASIC Training**, we'll continue looking at concepts that will help BASIC game programmers. The first topic is a return to issue 18's vector routines, and the second is an easy way to speed up your BASIC games.

Both of this issue's topics were taken from a letter I received from Harold L. Reed.

Vectors revisited.

Mr. Reed's letter begins:

Dear Tom,

I enjoyed your ***Basic Training*** *segment in the April issue of* ***ANALOG****. The follower routines you presented were very interesting, but as you said in your article, the routine that produces the best re-sults (Figure 6) has the drawback of being slow until the follower gets near the target. This occurs because the routine in Line 160, which determines the step size for X and Y, accomplishes the task by finding the total X and Y distances between the two points and dividing by 2 repeatedly, until both DELTAX and DELTAY are less than or equal to 1. So, when the points are far apart, the distances have to be divided by 2 many times in order to make them less than or equal to 1. The closer the points come to each other, the less times the dividing loop has to be repeated, and the faster the results appear.*

This can easily be solved. Since the objective is to scale both DELTAX and DELTAY so that the larger of them equals 1, simply determine which one is larger, divide the smaller by the larger, then set the larger equal to 1. This can be done by replacing Line 160 with the following:

```
160 IF DELTAX>DELTAY THEN DELTAY=DELTA
Y/DELTAX:DELTAX=1:GOTO 170
165 DELTAX=DELTAX/DELTAY:DELTAY=1
```

This speeds up the operation considerably, since only one calculation is now needed. However, it also generates an error when the follower and target come together. This occurs because the program doesn't check to see if it should stop until it calculates its next move. So, if the points are right on top of each other and the routine tries to calculate the next position of the follower, it ends up dividing by zero, which generates the error.

This can be corrected by moving Line 190 to Line 225. The end point check, which was formerly done after the next set of calculations was completed, is now done after each move is completed. The routine is now very fast.

Is my face *red*! When I originally wrote the algorithm for this follower routine, I was working with assembly language, which does not have true division. I translated the routine into BASIC too literally, and didn't stop to think that BASIC had a faster solution. In any case, Figure 1 shows the new, improved "FOLLOWER (VECTOR 1)" routine, courtesy of Harold J. Reed.

Figure 1.

```
10 REM *** FOLLOWER (VECTOR 1) ***
20 REM
30 GRAPHICS 6:COLOR 1
40 DIM XS(15),YS(15):FOR I=1 TO 15:REA
D X,Y:XS(I)=X:YS(I)=Y:NEXT I
50 DATA 0,0,0,0,0,0,0,0,1,1,1,-1,1,0,0
,0,-1,1,-1,-1,-1,0,0,0,0,1,0,-1,0,0
60 FX=0:FY=0
70 TX=80:TY=40
80 STIK=STICK(0)
90 TX=TX+XS(STIK)
100 TY=TY+YS(STIK)
110 PLOT TX,TY
120 XD=SGN(TX-FX)
130 YD=SGN(TY-FY)
140 DELTAX=ABS(TX-FX)
```

```
150 DELTAY=ABS(TY-FY)
160 IF DELTAX>DELTAY THEN DELTAY=DELTA
Y/DELTAX:DELTAX=1:GOTO 170
165 DELTAX=DELTAX/DELTAY:DELTAY=1
170 XV=DELTAX*XD
180 YV=DELTAY*YD
200 FX=FX+XV
210 FY=FY+YV
220 PLOT FX,FY
225 IF INT(FX)=INT(TX) AND INT(FY)=INT
(TY) THEN 240
230 GOTO 80
240 ? "GOTCHA!":END
```

•

CHECKSUM DATA
(See page 23)

```
10 DATA 420,253,28,350,327,808,992,514
,642,729,181,329,339,185,196,6293
160 DATA 146,886,580,589,820,828,130,7
65,505,441,5690
```

•

Faster execution made easy.

Continuing with our special "Harold J. Reed" installment of **BASIC Training**, here's something simple you can do that can increase the execution speed of your BASIC programs. Harold writes:

I once developed a rather large program and then developed a title screen to go with it. To avoid run-ning the entire program as the title screen was being debugged, I worked on it separately. I used a FOR/NEXT loop to slow down my plotting routine to the desired speed. But then, when it was just right and I added it to my main program, it ran much slower! If BASIC interprets one line of a program at a time, why would the length of the program have any effect on the speed of execution?

This is a very good question, and one which most programmers don't think about or even realize. But the fact is: code placed at the end of a BASIC program executes *slower* than code at the begin-ning! Let's find out why.

When a program is sitting in the computer's memory, BASIC only knows where the *first* line is located. In order to find the second line, BASIC goes to the first line and gets the pointer to the next line.

If BASIC needs to find the tenth line of a program, it must get the first line and find the pointer to the second line. It then looks at the second line for the pointer to the third line, and so on until it gets the line it's looking for. Imagine how much time is wasted looking for, say, the 400th line of a program!

Just so I could see how much time is lost when code is placed at the end of a program, I put a FOR-NEXT loop at the beginning of the BASIC code for **Retrofire**, which is roughly 225 lines long. When executed, the loop took about 24 seconds.

I then placed the loop at the end of the **Retrofire** program and executed it. This time, it took 99 seconds, over *four times* as long as it did at the beginning of the program!

A program will be slowed down any time code near the end is referenced by line number. For example, the statements GOTO 1000, GOSUB 1000, RESTORE 1000, etc. would all slow down the program if the line number referenced was at the end of the program. FOR/NEXT loops are also affected, since internally BASIC keeps track of the line containing the FOR statement.

If you write games in BASIC, it's a good idea to place one-time initialization routines or title screens at the end of the program. Keep often-used subroutines and the main control code at the beginning of the program. Remember, simply by organizing your code more carefully, you can increase the program's speed by several times!

Write on.

I'd like to thank Mr. Reed for sharing his insights with all the **ANALOG** readers. If you've got a question or observation, scribble it down on a postcard and send it to **BASIC Training**. There's no such thing as a "stupid" question, and you could help potentially thousands of other readers with the same problem.

Until next time, see if you can improve the performance of your old BASIC games by reorganizing the code. You may be pleasantly surprised. □

ConTEXT

16K Disk

by Vern L. Mastel

This program is written to run on any Atari computer with 48K of memory and at least one disk drive. **Letter Perfect** is the only Atari word processor that does not use the standard Atari file management system. Because of this, it is impossible to load **Letter Perfect** text files using another word processor, such as **Text Wizard**, or check the spelling using a standard spelling checker.

ConTEXT is a program which bridges this gap. It will take any text file prepared with **Letter Perfect** and reformat it into a conventional file that can be accessed with a word processor like **Text Wizard**. It will handle files up to 160 sectors in length (this can be increased) and works with one or two disk drives.

At its heart, the program is really quite simple. **Letter Perfect** uses a directory located from sectors 363 to 371 on the disk. **ConTEXT** uses a direct sector read routine to extract the file information and display it in file, starting sector and number of sectors format. Once the particular file to be converted is chosen, the text is read from the **Letter Perfect** file into a string for temporary storage. When the read process is complete, the string is then written back out to a new file on a standard Atari-formatted disk. This file can then be accessed by any conventional Atari word processor. The program offers options for using one or two disk drives.

The operation of the program is as follows:

Lines 10-80 do all of the initialization. There are two very short machine language subroutines used in **ConTEXT.** The first, INSTR$, calls the CIO get sector routine. The second, in OUTSTR$, calls the CIO record move routine. TRANSFER$ holds the text from the **Letter Perfect** file and can be adjusted in size to accommodate the average size of file used.

Lines 100-190 do the **Letter Perfect** directory read. The directory information is put into BUFFER$.

Lines 200-300 pull the individual file names, sizes and locations out of the information in BUFFER$.

Lines 310-480 handle the user input regarding which text file is to be converted.

Lines 490-640 do the work of reading the individual sectors of the **Letter Perfect** file and putting the text into TRANSFER$.

Lines 650-710 allow one- or two-drive option to be used for the output file.

Lines 720-780 call the CIO and pass the necessary values it needs to move TRANSFER$ to a standard Atari disk file.

Lines 790-830 handle prompting for a disk swap, if needed, upon completion of conversion and continuation or exit from the program.

Using ConTEXT.

The first thing to do when using **ConTEXT** is to copy the files to be converted onto a new **Letter Perfect** formatted disk. This is to insure that all of the sectors are in consecutive order for each file. **ConTEXT** is not smart enough to figure out where the right sectors are for a given file, if they are scattered about on a disk (a common situation on discs that are heavily edited). If you have two drives, the LP text disk will go into drive 1 after **ConTEXT** has been loaded into the computer. RUN the program and select the file to be converted. When you enter the starting sector and number of sectors, be sure to enter them correctly. An error can produce a totally scrambled output file, because the wrong sectors were read.

Letter Perfect has one very strange quirk which can cause a problem with conversion. An LP-formatted disk has sectors 8 through 55 reserved for some specific purpose, meaning that a 10-sector file will be written from sectors 3 to 7 and then from 56 to 61. Obviously, this file will not convert properly, because **ConTEXT** reads consecutive sectors. The solution is to save the file twice and ignore the first file. The program could also be rewritten to handle this situation, but I decided that the extra code was not justified. The destination disk for the converted file is a standard Atari-formatted disk. It can be a blank disk or one already containing text or program files. I prefer to keep converted files on their own disks. Once the converted file is written out to the destination disk, you are done — unless you wish to convert more files. Load in your **Text Wizard** or **Atari Writer** and proceed to use the new files.

One final note. Nearly all converted files will need some cleanup. Watch out for embedded control characters in the text and garbage at the very end. This garbage comes from the fact that rarely does a text file completely fill the last sector. Because **ConTEXT** reads complete sectors only, any "stuff" past the end-of-file in the last sector will be read in as well. It is this that produces the garbage displayed at the end of the text, which must be cleaned up. □

```
10 DIM INSTR$(5),DRIVE$(15),BUFFER$(12
8),OUTSTR$(7)
20 DIM A$(1),FILE$(12),TRANSFER$(20000
)
30 POKE 712,148:POKE 752,1
40 FOR X=1 TO 5:READ A:INSTR$(X)=CHR$(
A):NEXT X
50 DATA 104,32,83,228,96
60 FOR X=1 TO 7:READ A:OUTSTR$(X)=CHR$
(A):NEXT X
70 DATA 104,104,104,170,76,86,228
80 BUFFER$=" ":BUFFER$(128)=" ":BUFFER
$(2)=BUFFER$
90 POS=ADR(BUFFER$)
100 POSHI=INT(POS/256)
110 POSLO=POS-POSHI*256
120 POKE 772,POSLO
130 POKE 773,POSHI
140 FOR SECTOR=363 TO 371
150 SECTORHI=INT(SECTOR/256)
160 SECTORLO=SECTOR-SECTORHI*256
170 POKE 778,SECTORLO:POKE 779,SECTORH
I
180 POKE 770,82:POKE 769,1
190 IN=USR(ADR(INSTR$))
200 ? CHR$(125):POSITION 0,0:? "
 ";CHR$(8);"LETTER PERFECT DIRECTORY";
CHR$(10)
210 ? "FILE NAME      STARTING      N
UMBER"
220 INDEX=1
230 FOR ENTRY=1 TO 8
240 IF ASC(BUFFER$(ENTRY*16-10,ENTRY*1
6-10))=0 THEN 300
250 IF BUFFER$(ENTRY*16-15,ENTRY*16-15
)=" " THEN 300
260 ? BUFFER$(ENTRY*16-10,ENTRY*16);
270 POSITION 18,INDEX+1:? ASC(BUFFER$(
ENTRY*16-12))+256*ASC(BUFFER$(ENTRY*16
-11));" ";
280 POSITION 33,INDEX+1:? ASC(BUFFER$(
ENTRY*16-14))+256*ASC(BUFFER$(ENTRY*16
-13))
290 INDEX=INDEX+1
300 NEXT ENTRY
310 POSITION 2,10:? "    PRESS RETURN
FOR NEXT SECTOR":? "ENTER C TO CONVERT
 A DISPLAYED FILE";
320 INPUT A$:IF A$="C" THEN 370
330 NEXT SECTOR
340 ? CHR$(125):POSITION 8,10:? "
END OF DIRECTORY ":? "      PRESS RETUR
N TO BEGIN AGAIN"
350 ? "          ENTER E TO END";:INP
UT A$:IF A$="E" THEN END
360 GOTO 80
370 ? :? "FILE TO CONVERT====>";:INPUT
 FILE$
380 FOR X=1 TO LEN(FILE$):IF FILE$(X,X
)="." THEN FILE$=FILE$(1,X-1):GOTO 400
390 NEXT X
400 ? "INPUT STARTING SECTOR===>";:INP
UT SS
410 ? "INPUT NUMBER OF SECTORS=>";:INP
UT SECNUM
420 ? :? "FILENAME IS========>";FILE$
430 ? "STARTING SECTOR====>";SS
440 ? "NUMBER OF SECTORS==>";SECNUM
450 ? "IS THIS CORRECT Y/N";:INPUT A$:
IF A$="Y" THEN 480
460 POSITION 2,12:FOR X=1 TO 13:PRINT
CHR$(156);:NEXT X
470 POSITION 2,12:GOTO 370
480 POSITION 2,12:FOR X=1 TO 13:? CHR$
(156);:NEXT X
490 POSITION 2,13:? " CONVERTING FILE
";FILE$;" PLEASE WAIT"
500 TRANSFER$=""
510 MOVESIZE=(SECNUM-1)*128:IF MOVESIZ
E<128 THEN MOVESIZE=128
520 TRANSFER$=" ":TRANSFER$(MOVESIZE)=
" ":TRANSFER$(2)=TRANSFER$
530 POS=ADR(TRANSFER$)-128
540 FOR SECTOR=SS+1 TO SS+SECNUM-1
550 POS=POS+128
560 POSHI=INT(POS/256)
570 POSLO=POS-POSHI*256
580 POKE 772,POSLO:POKE 773,POSHI
590 SECTORHI=INT(SECTOR/256)
600 SECTORLO=SECTOR-SECTORHI*256
610 POKE 778,SECTORLO:POKE 779,SECTORH
I
620 POKE 770,82:POKE 769,1
630 IN=USR(ADR(INSTR$))
640 NEXT SECTOR:EFLAG=0:FOR ADDR=POS T
O POS+127:BYTE=PEEK(ADDR):IF BYTE=27 T
HEN EFLAG=1
644 IF EFLAG THEN BYTE=32
645 POKE ADDR,BYTE:NEXT ADDR
650 ? CHR$(125):POSITION 3,10:? "SEND
NEW FILE TO WHICH DRIVE D1-D2?":? :? "
          RETURN FOR D2"
660 ? :? "              ==>";:INPUT DRIV
E$
670 IF DRIVE$<>"D1" THEN DRIVE$="D2"
680 DRIVE$(3)=":":DRIVE$(4)=FILE$:DRIV
E$(LEN(DRIVE$)+1)=".CON"
```

```
690 ON DRIVE$(1,2)="D2" GOTO 720:? CHR
$(125):POSITION 8,10:? "PLEASE REMOVE
THE TEXT DISC"
700 ? "     AND INSERT AN ATARI FORMAT
TED":? "       DESTINATION DISC IN DRIV
E 1":?
710 ? "          PRESS RETURN WHEN READY"
;:INPUT A$
720 OPEN #1,8,0,DRIVE$:POS=ADR(TRANSFE
R$)
730 SIZE=LEN(TRANSFER$):SIZEHB=INT(SIZ
E/256):SIZELB=SIZE-SIZEHB*256
740 STARTHB=INT(POS/256):STARTLB=POS-S
TARTHB*256
750 POKE 852,STARTLB:POKE 853,STARTHB
760 POKE 856,SIZELB:POKE 857,SIZEHB
770 POKE 850,11
780 OUT=USR(ADR(OUTSTR$),16):CLOSE #1
790 ? CHR$(125):POSITION 10,8:? " CONV
ERSION COMPLETE ":? :?
800 ON DRIVE$(1,2)="D2" GOTO 820:? "
PLEASE REMOVE THE CONVERSION DISC":? "
   FROM DRIVE #1  AND REINSERT THE"
810 ? "       LETTER PERFECT TEXT DISC"
:?
820 ? "       PRESS RETURN TO CONTINUE"
:? "          ENTER E TO END";:INPUT
A$:IF A$="E" THEN END
830 GOTO 80
```

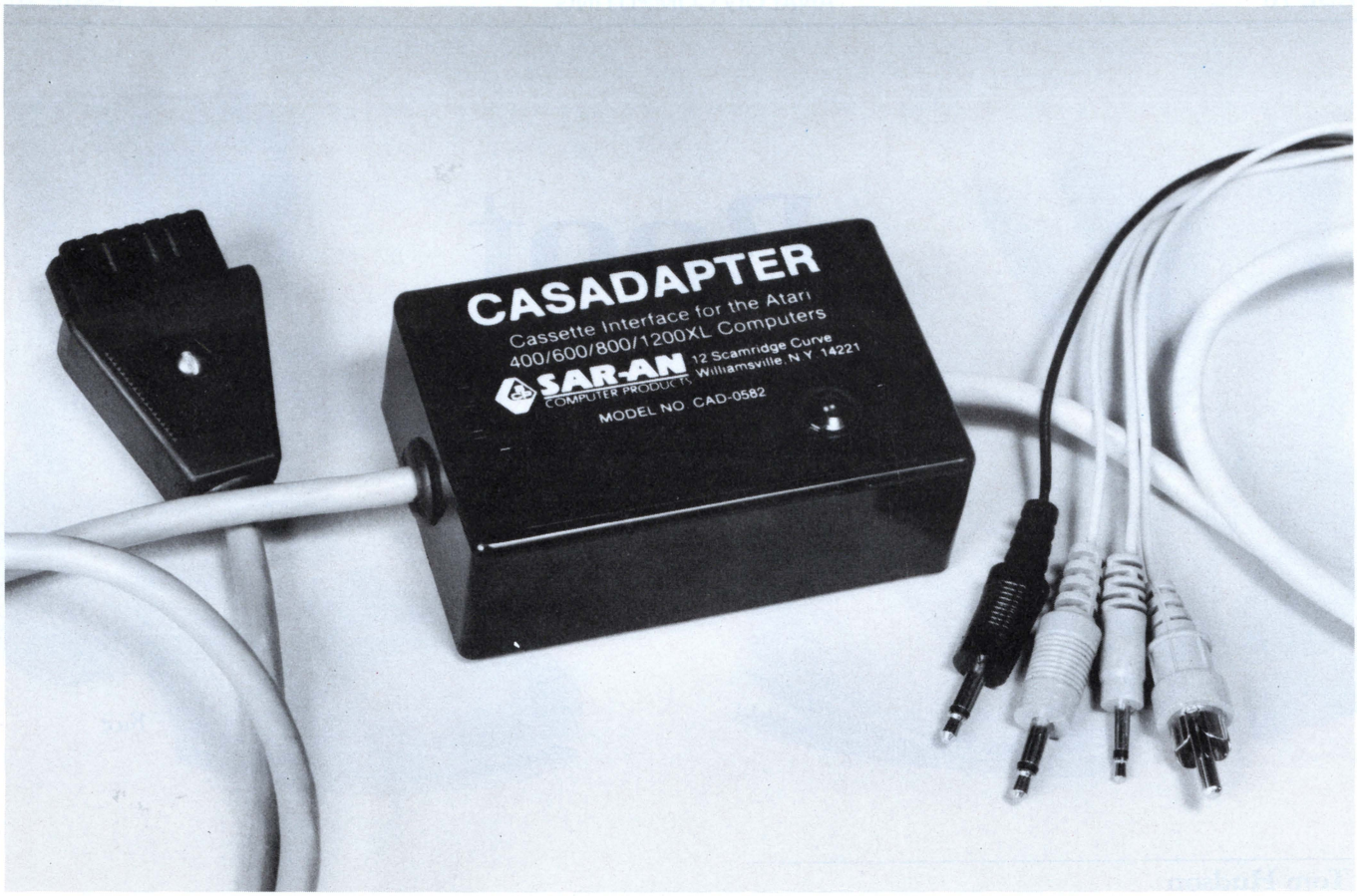
●

CHECKSUM DATA

(See page 23)

```
10 DATA 397,689,174,40,802,370,682,79,
922,229,195,583,569,431,129,6291
160 DATA 557,546,258,182,321,341,514,6
65,870,564,617,431,101,349,163,6479
310 DATA 327,956,367,869,105,516,858,4
68,789,174,328,902,483,637,951,8730
460 DATA 284,371,355,964,394,358,839,5
89,771,189,255,221,621,149,549,6909
610 DATA 538,250,174,805,640,262,143,5
77,551,681,181,51,507,343,419,6122
740 DATA 793,332,720,8,148,965,731,1,6
76,517,4891
```

●



CASADAPTER

THE ORIGINAL CASSETTE INTERFACE...

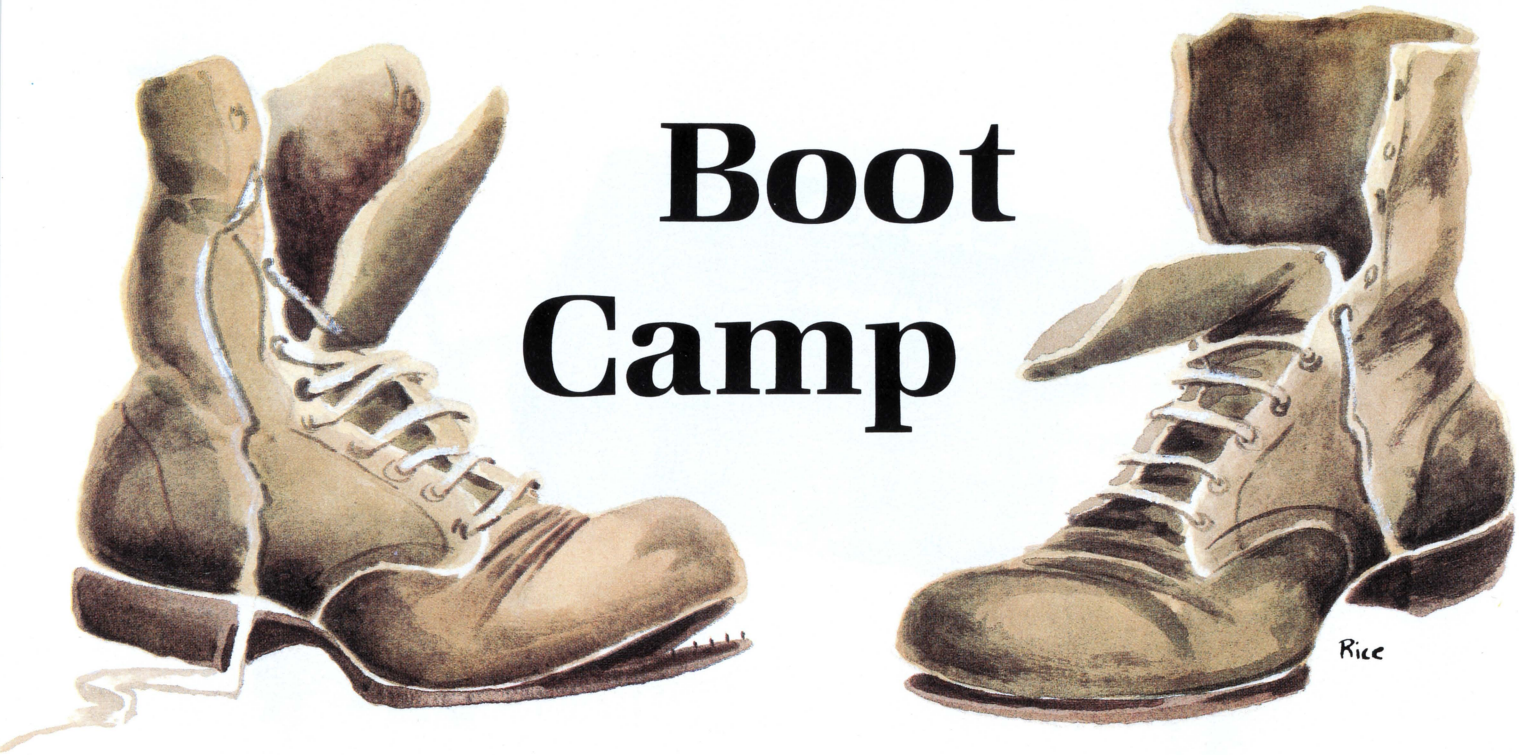
Allows you to use your own cassette recorder or stereo for loading and saving of programs. It makes your own home stereo or portable cassette player act just like an Atari 410 or 1010 recorder. CASADAPTER gives you motor control, audio, and data channels. The CASADAPTER plugs directly into the computer and requires no other interfaces. An L.E.D. shows the transfer of all data. CASADAPTER will work with all of the Atari brand computers.

$39.95

Add $2.50 shipping
Send check or money order
C.O.D. accepted
New York State Residents add 7% tax
®Atari is a trademark of Atari, Inc.

SCP SAR-AN COMPUTER PRODUCTS

12 Scamridge Curve
Williamsville, New York 14221
(716) 632-3441



Boot Camp

by Tom Hudson

Welcome back! As I mentioned last issue, there are only a few more 6502 instructions left for us to cover, and we'll talk about them in the next two installments. There are also a couple of instructions we're going to skip until later. They are for more advanced uses, and won't make much sense until you've got more experience with assembly language.

Several people have written lately, asking if we'll get into full-scale programs and using the Atari's powerful operating system. The answer: You bet! We're going to find out how to access the disk, cassette, graphics, keyboard, and just about anything else you'd like to hear about. We'll study routines for high-speed math, player/missile graphics, and more. If you've got a specific suggestion, drop me a postcard, and I'll add your idea to my topic file. **Boot Camp** is here not only to teach you what 6502 assembly instructions do, but how to apply them.

Two solutions.

Last issue, I asked you to write a program which multiplied the number 5 by 27. There is an almost infinite number of ways to do this, and I'll show you two of them now. Remember, these aren't the only possibilities, and, even though your solution may not be as efficient, getting the correct answer is what counts most.

Solution #1.

```
10        *=  $0600
20        CLD             ;BINARY MATH!
30        LDA #5          ;GET # TO MULT.
40        STA TIME51      ;SAVE # TIMES 1
50        ASL A           ;*2
60        STA TIME52      ;SAVE # TIMES 2
70        ASL A           ;*4
80        ASL A           ;*8
90        STA TIME58      ;SAVE # TIMES 8
0100      ASL A           ;*16
0110      CLC             ;CLEAR FOR ADD
0120      ADC TIME58      ;*24
0130      CLC             ;CLEAR AGAIN
0140      ADC TIME52      ;*26
0150      CLC             ;CLEAR AGAIN
0160      ADC TIME51      ;*27
0170      STA RESULT      ;SAVE # TIMES 27
0180      BRK             ;WE'RE DONE!
0190 TIME51 *=*+1
0200 TIME52 *=*+1
0210 TIME58 *=*+1
0220 RESULT *=*+1
0230      .END
```

Figure 1.

The first solution I'm going to cover is shown in Figure 1. This program uses the principle of breaking a multiply into "bite-sized" pieces, as shown last issue. In this case, I broke the multiply by 27 down into the following group of adds:

```
   (number * 16)
   (number *  8)
   (number *  2)
+  (number     )
-------------------
   (number * 27)
```

Let's step through the program in Figure 1 and see how it works.

Line 20 — clears the decimal mode. Always remember to be sure of the setting of the

decimal flag before doing any arithmetic.

Line 30 — loads the accumulator with the number 5. When the routine is finished, this number will be multiplied by 27 and stored in the memory location labeled RESULT.

Line 40 — stores the accumulator's contents in the memory location labeled TIMES! (5 * 1). We need to save this value for later, when we add the "bite-sized" pieces together.

Line 50 — shifts the accumulator contents left one bit, multiplying it by two.

Line 60 — saves the accumulator (now 5 * 2) in the location TIMES2. This value is also needed for our final result.

Line 70 — shifts the accumulator left one bit again, leaving the accumulator with the value 5 * 4.

Line 80 — performs another left shift on the accumulator. The accumulator now contains 5 * 8.

Line 90 — saves the accumulator's contents in the location TIMES8.

Line 100 — performs a final left shift on the accumulator, leaving the accumulator with the value 5 * 16. At this point, we have all the "bite-sized" pieces we need to get our answer, and are ready to add them up.

Line 110 — clears the carry flag for the first add in the group. Remember, this is a necessary instruction before any single-byte addition.

Line 120 — adds the accumulator (5 * 16) to TIMES8 (5 * 8), leaving the result (5 * 24) in the accumulator for the next add.

Line 130 — clears the carry for the next add.

Line 140 — adds the accumulator (5 * 24) to TIMES2 (5 * 27), with the result (5 * 26) left in the accumulator.

Line 150 — clears the carry again, for the final addition operation.

Line 160 — adds the accumulator (5 * 26) to TIMES1 (5 * 1), leaving the accumulator holding the final value, 5 times 27!

Line 170 — saves the final answer in the location labeled RESULT.

Line 180 — BREAKs the execution of the program. At this point, you can check the location RESULT to be sure it contains 5 * 27, or 135 ($87 hex).

Lines 190-220 — reserve one byte for each of the four data areas used by the program.

Solution #2.

The second solution I decided to show is a modification of the first technique. In this program, I decided to break the multiply down into smaller pieces again, but structure it so that subtracts are used instead of adds:

```
      (number * 32)
      (number *  4)
    - (number     )
    ----------------
      (number * 27)
```

As you can see, we get the same result as with adds, but with only three math operations instead of four. Figure 2 shows the 6502 code necessary to implement this method.

```
10        *=  $0600
20        CLD             ;BINARY MATH
30        LDA #5          ;GET # TO MULT.
40        STA TIMES1      ;SAVE # TIMES 1
50        ASL A           ;*2
60        ASL A           ;*4
70        STA TIMES4      ;SAVE # TIMES 4
80        ASL A           ;*8
90        ASL A           ;*16
0100      ASL A           ;*32
0110      SEC             ;SET FOR SUBTRACT
0120      SBC TIMES4      ;*28
0130      SEC             ;SET AGAIN
0140      SBC TIMES1      ;*27
0150      STA RESULT      ;SAVE # TIMES 27
0160      BRK             ;ALL DONE!
0170 TIMES1 *=*+1
0180 TIMES4 *=*+1
0190 RESULT *=*+1
0200      .END
```

Figure 2.

Let's walk through this program and see what's going on.

Line 20 — clears the decimal mode for binary arithmetic. I can't overemphasize the importance of knowing the status of the decimal mode flag. If you're in doubt, SET or CLEAR it as needed.

Line 30 — loads the accumulator with the number 5. When this program is finished, the number 5 will be multiplied by 27.

Line 40 — saves the contents of the accumulator in the location labeled TIMES1, for later use.

Line 50 — shifts the accumulator left 1 bit, multiplying it by 2.

Line 60 — shifts the accumulator left again, leaving the accumulator with the value 5 * 4.

Line 70 — saves the contents of the accumulator (5*4) in the memory location TIMES4.

Line 80 — shifts the accumulator left again, leaving the value 5 * 8 in the accumulator.

Line 90 — performs another left shift. At this point the accumulator contains 5 * 16.

Line 100 — shifts the accumulator left a final time. The accumulator now contains the value 5 * 32. We are now ready to perform the subtract operations as shown above.

Line 110 — sets the carry flag for the first subtract operation. Remember, the carry flag should always be set before a single-byte subtract to insure correct results.

Line 120 — subtracts the value TIMES4 (5 *

4) from the accumulator (5 * 32), leaving the accumulator containing the value 5 * 28.

Line 130 — sets the carry flag for the next subtract.

Line 140 — subtracts the value TIMES1 (5 * 1) from the accumulator (5 * 28), leaving the accumulator with the value 5 * 27!

Line 150 — saves the answer in the location labeled RESULT.

Line 160 — stops the program's execution with the BRK instruction. At this point, you can verify that the location RESULT (and the accumulator) contain 5 * 27, or 135 ($87 hex).

Lines 170-190 — reserve one byte for each of the three data fields used by the program.

Obviously, these are just two of the thousands of solutions possible for this problem. If you've got a different approach, I'd like to see it. Just send your programs to **Boot Camp**, in care of **ANALOG**.

Stacking the deck.

The last topic we're going to cover before going on to bigger and better things is the 6502 *stack*. This is an important feature of the 6502, as it allows us to write subroutines. Since the stack concept is very important, we're going to cover it in detail starting this issue, and finish it with assembly examples next time. Let's get started finding out what the stack is and how it works.

The 6502 reserves 256 bytes of memory from $0100-01FF (also called page 1) for a temporary storage area. We call this area the *stack*. This area is automatically maintained for the 6502, but we can use it for short-term storage, too.

We call the stack a "last-in, first-out" structure. The last number placed on the stack is always the first to be pulled off. A good way to remember this is to think of a stack of pancakes. When you pile them up, the last one put on the stack is on top. When you take them off one at a time, the last one you put on comes off first. Using this analogy, the computer could keep track of 256 pancakes, each with a number written on it.

The computer keeps track of the stack's contents by using the Stack Pointer register inside the 6502. This pointer ranges from $00-FF. When the stack pointer contains $00, it is pointing to the memory location $0100. When it contains $FF, the location $01FF is indicated.

(continued on page 80)

Interestingly, the stack works backwards from the way we would expect. When the stack is empty, the stack pointer is set to $FF. Figure 3 shows an empty stack.

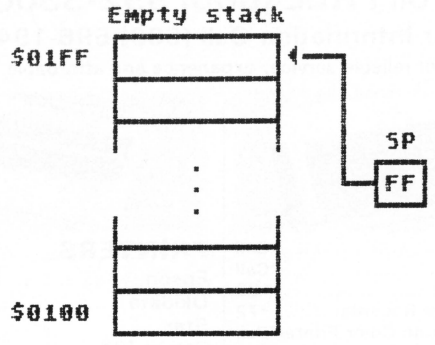


Figure 3.

As the stack is filled with more and more values, the stack pointer is decremented, pointing to lower areas of page 1. When completely filled, the stack pointer will contain $00, as shown in Figure 4.

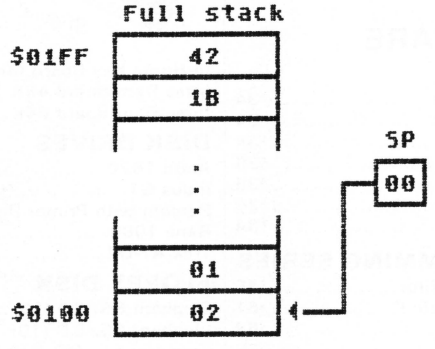


Figure 4.

Since the computer has only reserved 256 bytes for a stack, there are obviously limitations in its use. If the stack is filled with too many values, the stack pointer will "wrap around" back to $FF and begin wiping out earlier stack entries! There is no "error message" for this, so you must be careful when working with the stack.

When entries are removed from the stack, the process is reversed. As each byte is pulled off the stack, the pointer is *incremented*, pointing to progressively higher locations of the stack.

How subroutines work.

In BASIC, subroutines are easy to write. You simply set up the necessary BASIC code, put a RETURN instruction at the end of it, and call it with the GOSUB statement whenever you need it. The subroutine code is performed, and BASIC resumes execution at the next statement after the GOSUB. Neat, huh?

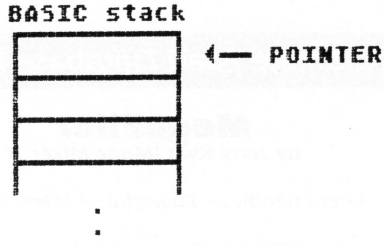
In order for a BASIC subroutine to work, the computer has to know how to get back to the instruction after the GOSUB. It does this by using a stack. Let's look at a simplified example of how a BASIC subroutine is executed.

```
10 GOSUB 100
20 END
100 GOSUB 200
110 RETURN
200 A=A+1
210 RETURN
```

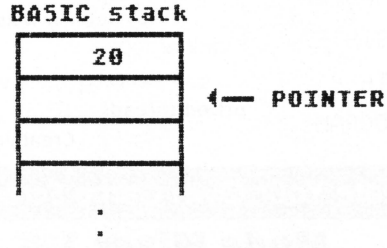
Figure 5.

Figure 5 is a short BASIC program using the BASIC subroutine statements, GOSUB and RETURN. We're going to step through it and watch what happens to the BASIC stack, a special area similar to the 6502 stack.

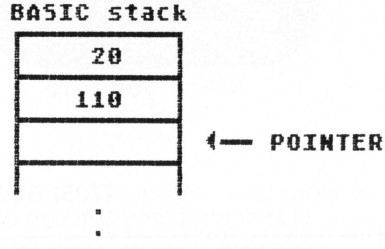
Before execution — The stack is empty, and the stack pointer is pointing to the first available position.



Line 10 — The GOSUB to Line 100 is executed. First, the computer finds the next statement after the GOSUB. The next statement is in Line 20, so the computer *pushes* that line number onto the first location on the stack, and changes the stack pointer to point to the next available location. Execution then proceeds at Line 100. At this point, the stack looks like:

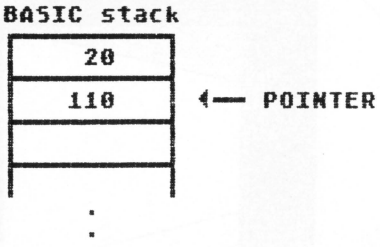


Line 100 — This line executes a GOSUB to Line 200. The next statement after this GOSUB is Line 110, so this number is placed on the stack, and the stack pointer is advanced to the next available position. Execution continues at Line 200. The stack now looks like:



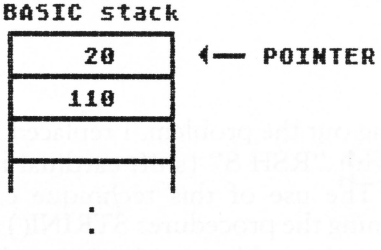
Line 200 — The computer adds one to the variable A. The stack is not affected.

Line 210 — The computer encounters a RETURN statement. At this point, the computer *increments* the stack pointer, like so:



Now the computer takes the line number 110 from the stack. As you can see, the computer can now go back to the instruction after the last GOSUB. Execution continues at Line 110.

Line 110 — Another RETURN is encountered, and the stack pointer is incremented again. Now the stack looks like this:



The computer gets the line number from the stack and completes the RETURN by resuming execution at Line 20.

Line 20 — This line terminates execution with the END statement. The stack is back to its original condition, with the pointer indicating the first stack location. The line numbers are still in the stack itself, but since the stack pointer no longer points to them, they are no longer active. They will be wiped out by new stack entries.

Now do you see how the stack works? It's a great way to handle subroutines, where the computer must be able to find its way back to the code which called the subroutine.

Until next time.

If you think **Boot Camp** looks more like **Basic Training** this issue, hold on! I wanted to explain the subroutine process in a language you're familiar with, like BASIC. Next issue we'll examine the operation of the 6502 subroutine process, and learn how to use the stack for our own programs.

```
10 GOSUB 10
20 END
```

Figure 6.

Until we meet again, here's a little program to get you thinking. Type in the BASIC program in Figure 6 and RUN it. It may take a while, but something will happen, and I want you to see if you can find the cause. Use the stack illustration method I used in the BASIC example to get the answer.

Also, if you haven't already, try to find more alternate methods for multiplying 5 by 27! □

Address all letters to:

BOOT CAMP
P.O. Box 23
Worcester, MA 01603

STARS 3-D in Action!

16K Disk

by Donald E. Glover

I was looking for something to do with my shiny new Action! cartridge when I ran across the article **Stars 3-D** by Craig Patchett in **ANALOG** No. 16. 16. To become familiar with the new language, I decided to translate this demonstration program into Action!, a job I thought would take one or two hours. The task eventually took much longer, due to a number of strange quirks associated with the Action! language. I hope this discussion of my problems will save other Action! programmers some hair pulling and nail chewing.

My first task was to find a place for the display list (DLIST) and screen memory (STRLIN). I wanted to put them in a safe location, while allowing easy access from Action!-generated code and in-line machine code. I finally decided to put them in Action! arrays whose starting addresses were defined such that the display list and screen memory started on 1K boundaries in high memory. (The Atari cannot easily deal with a display list which crosses a 1K boundary or screen memory which crosses a 4K boundary.)

Calculations to generate the display list required that the address of screen memory be divided by 256 to obtain the high byte of the address. Performing this division on addresses greater than 32767, unfortunately, gives the wrong answer, since Action! multiplications and divisions always assume they are acting on signed numbers. Try typing:

```
X PRINTCE(32768/256)
```

in the Action! monitor and see what you get. After figuring out the problem, I replaced the division by 256 with "RSH 8" (shift cardinal number right 8 bits). The use of this technique can be seen by examining the procedures STRINI() and DLSINI().

The next problem was to insert the addresses of the arrays STRTPH, STRTPL, and STRPOS into the machine language procedure SCROLL(). My initial attempt to do this involved inserting the address during the compilation phase. Using this method, the first instruction in the procedure SCROLL() would be:

```
$BO STRTPL ;LDA STRTPL, X
```

To my horror, the addresses of arrays compiled into the code by this technique frequently (but not always) differed from those observed after compilation. Apparently, the addresses of arrays change during the compile phase, and the compiler cannot modify addresses inserted into machine code. The solution was to "POKEC" the addresses into the machine language routines during run time [see the procedure MAIN()].

I believe everything else in the listing is understandable, because I kept the names of all routines and most of the comments the same as those in the original assembly language listing. A word of warning: this program is designed to work with a machine having 48K of memory. If your machine has less memory, you will have to change the starting address of the arrays DLIST and STRLIN. The place to do this is clearly marked in the listing.

Before finishing, I should mention another couple of Action! peculiarities.

Action! Listing

```
; ***********************
; *  STARS 3D DEMO      *
; *  BY CRAIG PATCHETT  *
; *  ACTION! VERSION    *
; *     BY DON BLOVER   *
; ***********************
;system equates
CARD sdmctl=$22F,
     nmien=$D40E,
     random=$D20A,
     vdslst=$200,
     colpf0=$D016,
     colbk=$D01A,
     xitvbl=$E462;
BYTE index ;used to index into strcol
CARD indrct=$D6 ;pointer for indirect addressing
BYTE ARRAY timary=[8 8 6 6 5 5 3 3 2 2 1 1] ; timers for scrolling
timarycopy=$22 $24 $26 $28 $2A $2C $2E $22], ; star colors
; **careful-requires a 40k machine to work
dlist(600)=35840, ;display list
strlin(768)=36864, ;starlist(screen) memory
; ***********************************
strtpl(16), ;addresses of beginning of
strtph(16), ;each star line
strpos(16), ;position of star on line
strcol(192), ;color of each line
CARD ARRAY strtp(16) ;temp; hold addresses to be
                     ;transferred into strtpl strtph
PROC setvbv=$E45C(BYTE command,vblankhigh,vblanklow)
;***********************************
PROC SCROLL(BYTE dummy,starindex) ; star scroll routine
; ACTION! is too much trouble to use here with all the bit twiddling.
[ $8D $FFFF ;LDA STRTPL,X
  $85 INDRCT ; STA INDRCT
  $8D $FFFF ;LDA STRTPH,X
  $85 INDRCT+1 ;STA INDRCT+1
  $BC $FFFF ; LDY STRPOS,X
  $B1 INDRCT ; LDA (INDRCT),Y
  $0A ;ASL
  $0A ;ASL
  $91 INDRCT ;STA (INDRCT),Y
  $90 $0F ; BCC SCRBR1
  $A9 $01 ; LDA #1
  $88 ; DEY
  $C0 $FF ; CPY #255
  $D0 $02 ; BNE SCRBR2
  $A0 $2F ; LDY #47
;SCRBR2
  $91 INDRCT ; STA (INDRCT),Y
  $98 ; TYA
  $9D $FFFF ; STA STRPOS,X
;SCRBR1
]
RETURN
;***********************************
PROC cntdwn() ; timer routine
;***********************************
  BYTE starcounter
  IF timer(starcounter)=0 THEN
     scroll(0,starcounter) ; scroll if ready
     timer(starcounter)=timary(starcounter) ; reset timer
  FI
  OD
RETURN
;***********************************
PROC vblank() ; vblank routine
;***********************************
  cntdwn() ; back to scan line zero
  index=0
  [
  $4C xitvbl ; jmp xitvbv
  ]
RETURN ; never gets here
;***********************************
PROC strini() ; initialize stars
;***********************************
  BYTE counter
  ; set up screen memory address tables
  ; first use dummy array for simplicity
  FOR counter=0 to 15
  DO
     strtp(counter)=strlin+48*counter
     strpos(counter)=0 ;all stars start at pos. zero
  OD
  ; now transfer addresses to byte arrays
  FOR counter=0 to 16
  DO
     strtpl(counter)=strtp(counter)&255
     ; cannot divide numbers>32767 and get correct results
     strtph(counter)=strtp(counter) RSH 8
  OD
  ; clear star memory
     ZERO(strlin,768)
  ; give each line a star
  FOR counter=0 to 15
  DO
     strlin(48*counter)=64
  OD
  RETURN
; ***********************************
PROC dlsini() ; initialize display list
  BYTE startype,offset,counter
  CARD TEMP
  ; do for each scan line
  dlist(0)=$70 dlist(1)=$70 dlist(2)=$F0
  FOR counter=0 to 191
  DO
     dlist(3*(counter+1))=$CE ; graphics 7+
     ; now pick star type
     startype=RAND(8)
     ; set color for each line
     strcol(counter)=mancol(startype)
     ; randomize offset into memory
     offset=RAND(48) ;offset into line
     ; now put address of line into display list
     dlist(3*(counter+1)+1)=(strtp(2*startype)+offset)&255
     ; cannot divide numbers>32767 and get correct results
     dlist(3*(counter+1)+2)=(strtp(2*startype)+offset) RSH 8
  OD
  ; now finish up display list
  dlist(579)=$41 ; jump instruction
  dlist(580)=dlist&255
  dlist(581)=dlist RSH 8
  sdlstl=dlist ;tell antic where display list is
  sdmctl=$23 ;wide screen
RETURN
; ***********************************
PROC dli() ; display list interrupt routine
; too little time to even think about using ACTION!
  [
  $AE INDEX ; LDX INDEX
  $BD $FFFF ;LDA STRCOL,X
  $8D WSYNC ; STA WSYNC
  $8D COLPF0 ; STA COLPF0
  $EE INDEX  ; INC INDEX
  $40 ;RTI
  ]
RETURN
;*****************************************************
PROC  MAIN()
  ; fix up ml routines
  ; by poking in addresses of various arrays
  POKEC(dli+4,strcol)
  POKEC($scroll+7,strtpl)
  POKEC($scroll+12,strtph)
  POKEC($scroll+17,strpos)
  POKEC($scroll+40,strpos)
  POKEC(DLI+1,@index)
  strini() ;set up stars
  dlsini() ;set up display list
  color4=0 ;set background color
  SETVBV(7,vblank/256,vblank&255) ; set up vblank
  vdslst=dli  nmien=192 ; get dli's going
  ; now just let things run
  DO

  OD
RETURN
```

1. Negative FOR loops do not work. Try:

```
FOR COUNTER=5 TO 0 STEP -1
```

in a sample test procedure. It won't work.

2. You cannot initialize a variable to a negative number.

TEMP1=[-1] will not work

However

temp1=[65535]

will accomplish the same thing.

3. The example on page 123 of the manual doesn't work (at least with Version 3.5 of Action!). PRINTCE(rec. idnum) prints the wrong answer. For some reason, PRINTCE(rec. idnum*1) gives the right answer.

I suspect Action! will never be used to write commercial arcade-style games, because it is just not as efficient at "bit-twiddling" as machine code (try writing the procedure scroll() in Action!). It also does not produce code as compact as that produced by a good assembly language hacker, a definite consideration when trying to stuff a game in an 8K or 16K ROM cartridge. Nevertheless, I feel that the language (perhaps with the help of a few machine language routines) can be used to write games similar in quality to the machine language games found in **ANALOG** in a much shorter time than usually required. Games written in Action! would also be easier for novices to analyze and understand. □

•

Lyco Computer Marketing & Consultants

TO ORDER CALL US

TOLL FREE 800-233-8760
In PA 1-717-327-1824

PRINTER INTERFACING
PRINTER PAPER AVAILABLE
Available for IBM PC, Apple, Atari, Vic 20 & Vic 64

SAVE on these in-stock PRINTERS

LETTER QUALITY

SMITH CORONA TP2...$449.00

DIABLO 630 ..$1719.00

ALPAHCOM 42 .....$89.00
ALPHACOM 81 .....$129.00
NEC 8023 .....$369.00
NEC 8025 .....$699.00
NEC PC-8200 COMPUTER .....$CALL

EPSON

RX-80 .....$SAVE$
RX-80FT .....ON
FX-80 .....In-Stock
FX-100 .....EPSON
MX-80FT .....PRINTERS
MX-100 .....$$CALL$$

MANNESMANN TALLY

SPIRIT 80 .....$CALL
MT 160L .....$CALL

OKIDATA

80 .....$SAVE$
82A .....CALL for
83A .....LOWEST
84 .....PRICES
92 .....on these
93 .....In-Stock
PACEMARK 2350...PRINTERS

ATARI 850 REPLACEMENTS IN-STOCK

CITOH

GORILLA GX100 .....$179.00
PROWRITER 8510 ...$339.00
PROWRITER II $659.00
8600 .....$1025.00
STARWRITER $1099.00
PRINTMASTER $1499.00

STAR MICRONTICS

GEMINI 10X $269.00
GEMINI 15X .....$CALL
DELTA 10 .....$479.00

MODEMS

ANCHOR MARK I ..$79.00
ANCHOR MARK II..$79.00
HAYES SMART ...$239.00
HAYES MICRO II $309.00
Micro Bit
MPP-1000 .....$129.75
NOVATION
CAT $144.00
D-CAT $155.00
J-CAT $115.00
APPLE CAT II $279.00
212 APPLE CAT ..$589.00

MONITORS

Sakata Color $229.00
Amdek Color I $275.00
Amdek 300 Green $149.00
Amdek 300 Amber $149.00
Gorilla Green $99.00

PARKER BROTHERS

Tutankham R.....$33.75
Super Cobra R $33.75
Astro Chase R.....$33.75
Frogger R $33.75
QBert R $33.75
Popeye R.....$33.75
Risk R $42.75
Chess R $42.75

SPINNAKER

Story Machine R $26.75
Face Maker R.....$24.75
Kinderomp R.....$20.75
Fraction Fever R.....$24.75
Delta Drawing R.....$26.75

SSI

Battle of Shilo C/D.....$26.75
Tigers in the Snow C/D....$26.75
Battle for Normandy C/D ..$26.75
Knights of the Desert C/D .$26.75
Cosmic Balance C/D $26.75

ON-LINE

Frogger $24.95
Wizard & Prin $26.95

ROKLAN

Wizard of War.....$29.75
Gorf $29.75
Delux Invader $27.95

BIG 5

Miner 2049.....$32.75

APX 3R Math..... $19.95
Typo Attack $24.95
Family Budget $19.95
F. Cash Flow..... $19.95

EASTERN HOUSE

Monkey Wrench 2 .....$52.75

BRODERBUND

Bank Street Writer D..... $49.75
AE D.....$24.75
Apple Panic D.....$23.75
Choplifter ROM $32.75
David's Midnight.....$24.75
Stellar Shuttle C/D.....$18.75
Ft. Apocalypse.....$24.75

SYNAPSE

BLUE MAX C/D $24.75
Ft. APOCALYPSE C/D...$24.75
PHAROAH'S CURSE C/D...$24.75

DUST COVERS

800 .....$3.99
400 .....$3.99
1200.....$3.99
410 .....$3.99
810 .....$3.99
1050.....$5.99
PROWRITER $5.99
GEMINI 10X.....$5.99
PERCOM DISK $5.99

ALIEN GROUP

Voice Box 2 . $99.75

DON'T ASK

Sam $41.75
Abuse $15.95
Teleatri.....$27.95
Poker Sam . .$24.95

PACMAN..... $29.75
DONKEY KONG $29.75
DIG DUG..... $29.75
DEFENDER.... $29.75
CENTIPEDE ... $29.75

EPYX

GATEWAY TO ASPHI R.....$28.75
JUMPMAN JR R $28.75
PIT STOP R $28.75
GATEWAY TO

ATARI
Computers for people.

600XL ...$CALL
800XL.....for
1400XL...Lowest
1450 .....Prices

1020 PRINTER.....NOW
1025 PRINTER.....IN
1027 PRINTER .....STOCK
1050 DISK DRIVE ....$SAVE$
1010 RECORDER.....$74.75

DEADLINE ..$34.75
ENCHANTER ..$34.75
INFIDEL .$34.75
PLANETFALL ..$34.75
STAR CROSS ..$34.75
SUSPENDED ..$34.75
WITNESS ...$34.75
ZORK I ...$34.75
ZORK II ...$34.75

BLANK DISKETTES

ELEPHANT

Single Side SD (10).....$17.75
Single Side DD (10).....$21.75
Double Side DD (10).....$26.75

MAXELL

MD I (10).....$28.75
MD II (10) $38.75

CERTRON CASSETTES

CC-10 12 for $15.99
CC-20 12 for $17.99

INNOVATIVE CONCEPTS

Disk Storage (holds 10)....$4.95
Disk Storage (holds 15)....$9.95
Disk Storage (holds 50)....$26.95

TRAK DISK DRIVES

AT-D1 .....$379.00
AT-D2 .....$399.00
PRINTER CABLE $22.95
Software for ATD-2 ....$22.95

RANA DISK DRIVE

COMPUTER CARE

BIB
5¼ DISK DRIVE CLEANER.....$12.75
COMPUTER CARE KIT $19.75

HARD DISK DRIVES for APPLE IBM-PC

5MEG.....$1349.00
10MEG ...$1599.00
15MEG ...$1999.00
20MEG ...$2359.00
*Add $ 30.00 for TRS 80 Drives

TEXAS INSTRUMENT
Disk Drive...$245.00

PERCOM

FOR ATARI COMPUTERS

AT88S1 ....$329.00
AT88S2 ....$535.00
AT**S1PD...$439.00
RFD40S1...$399.00
RFD40S2...$675.00
RFD44S1...$449.00
AT88 doubler

MasterCard **TO ORDER** VISA

CALL TOLL FREE
800-233-8760
Customer Service 1-717-327-1825

or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

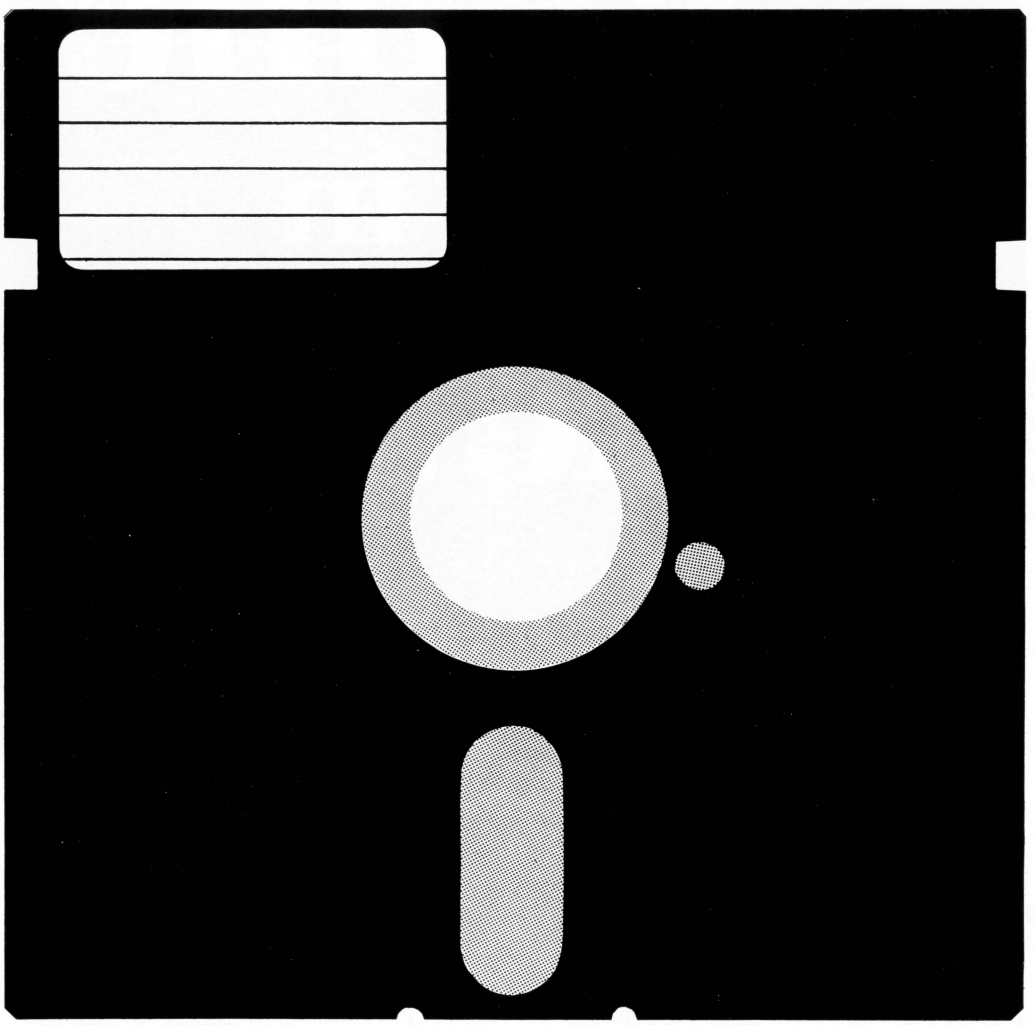
POLICY

In-stock items shipped within 24 hours of order. Personal checks require four weeks clearance before shipping. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. PA residents add sales tax. All products subject to availability and price change. Advertised prices show 4% discount offered for cash, add 4% for Master Card or Visa. DEALER INQUIRIES INVITED.

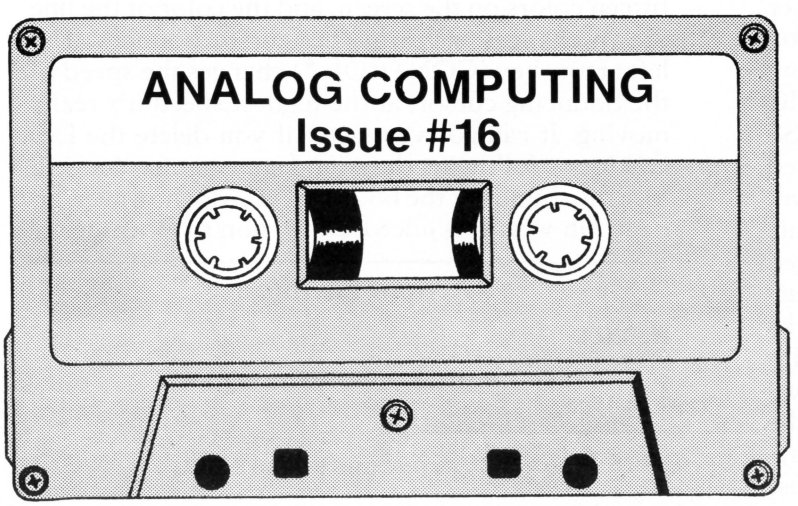
CIRCLE #149 ON READER SERVICE CARD.

DO YOU SUBSCRIBE TO ANALOG COMPUTING ON DISK OR CASSETTE?

. . . if not, then you should. Since issue 1, **ANALOG**'s cassette and disk subscriptions have eliminated the need for you to spend hours typing in programs from the magazine. All of the programs in the magazine are on the cassette and disk versions.



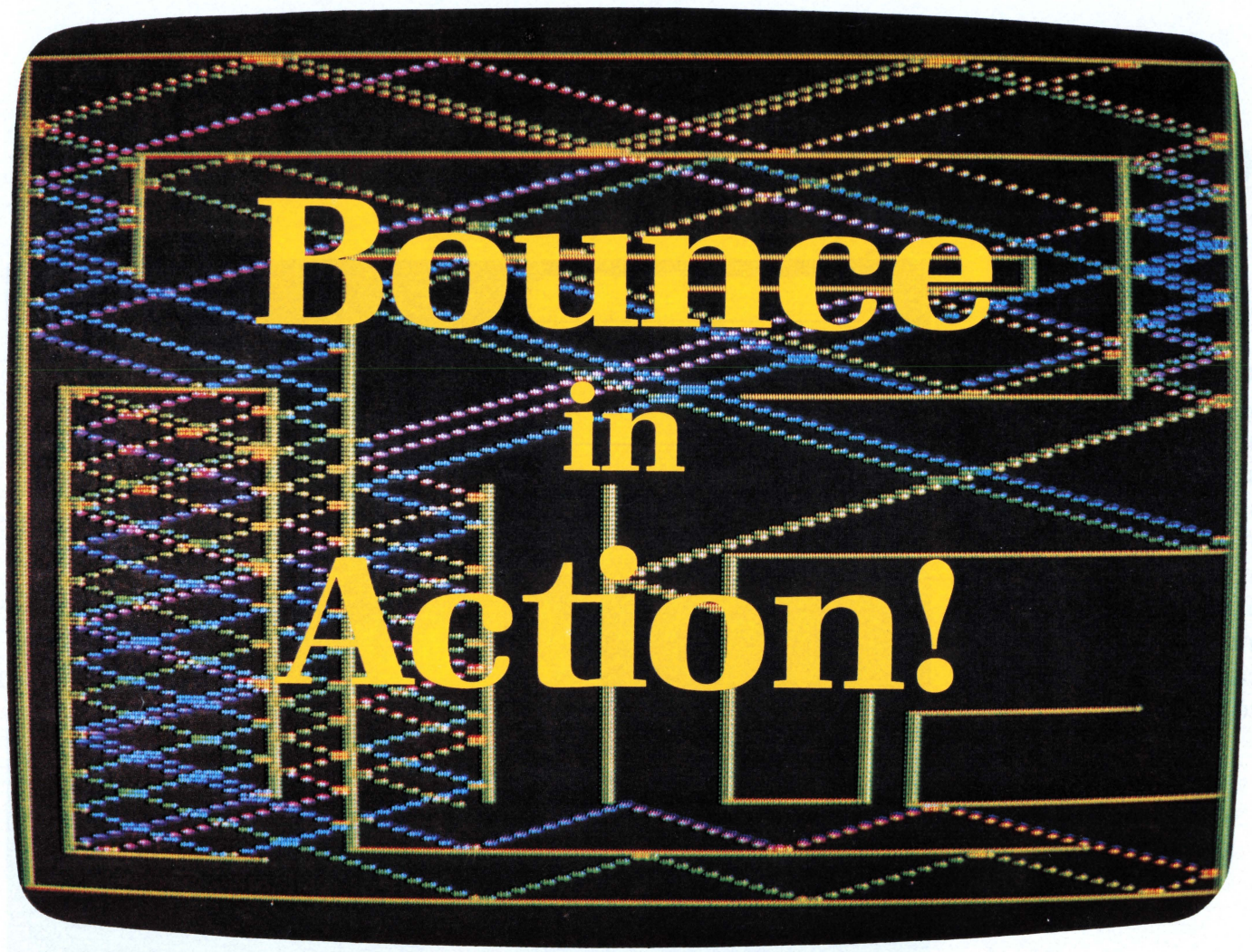
1 year (12 issues) on cassette, $90.00, disk, $130.00; ½ year (6 issues) cassette, $48.00, disk, $72.00.



To subscribe on cassette or disk, send check or money order to:

CASSETTE SUBSCRIPTION
or
DISK SUBSCRIPTION
P.O. BOX 615
HOLMES, PA 19043

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444



24K Cassette or Disk

by David Plotkin

Bounce, written by Joel Gluck and published in **ANALOG** issue 15, was a lot of fun to play with, just as Joel predicted it would be. The obvious enhancements that sprang to mind included a higher resolution screen and multiple colors. Unfortunately, higher resolution (and more than four colors) means more points to draw, and BASIC slows to a crawl. Fortunately, Action! from OSS presents an alternative, so I translated and modified the program into Action! Try punching it in; I think you'll agree that the color patterns and dynamic "ball" are fascinating to watch. To use this updated version of **Bounce**, you must have the Action! cartridge installed in your Atari. The program works pretty much like the original: You draw "walls" with your joystick, then hit the space bar to start the ball bouncing. Hitting the space bar again stops the bounce, so you can draw more walls with your joystick, or erase by pushing the fire button. If you press the ESCAPE key while the bouncing is stopped, you will return to the menu screen to review the commands. The program uses Graphics 11, so there are fifteen colors on the screen, and the color of the line drawn changes each time the cursor bounces. The left arrow key (CONTROL *) changes the speed of the bouncing cursor; at the highest speed it's really moving. It can go even faster if you delete the DO OD loops following the sound statements. You will lose the sounds of the bounce if you do, however. So have fun with this juiced-up version of **Bounce**. □

Action! Listing.

```
MODULE

; BOUNCE from ANALOG Magazine
; Issue #15
; in GTIA Mode 11

BYTE key=764,x,y,console=53279,
     attract=77
```

```
CARD ctr
INT A,B

PROC wallchex()

IF x)78 THEN x=78 FI
IF y)190 THEN y=190 FI
IF x(1 THEN x=1 FI
IF y(1 THEN y=1 FI
RETURN

PROC menu()

PrintE("BOUNCE from Analog Issue #15")
PrintE("       in GTIA mode 11")
PrintF("%E* Use stick to draw walls,%E")
PrintF("* Hold trigger to erase,%E")
PrintF("* Hit ESC to clear screen,%E")
PrintE("* Hit SPACE to bounce.")
PrintE("* Arrows  control ball speed")
Print("Press any key to continue.")
key=255
While key=255 Do Od
Key=255
RETURN


PROC drawscreen()
BYTE curs=752
Graphics(0)
curs=1
Menu()
Graphics(11)
curs=1
SetColor(4,0,4) ;SetColor(4,0,0)
color=15
Plot(0,0)
DrawTo(79,0)
DrawTo(79,191)
DrawTo(0,191)
DrawTo(0,0)
RETURN


PROC flash()
color=9
Plot(x,y)
FOR ctr=0 to 300 DO OD
color=0
Plot(x,y)
FOR ctr=0 to 300  DO OD
RETURN


PROC bounce()
BYTE fate=53770,L=[0],PA,PB,G,
     kolor=[1],time=[32]

color=9
A=1
B=1
Plot(x,y)
DO
    IF key=33 THEN key=255 RETURN FI
    WHILE Locate(x+A,y+B)(15 Do
        color=kolor
        Plot(x,y)
        x==+A
        y==+B
        wallchex()
        color=9
        Plot(x,y)
        L==+1
        FOR ctr=0 to 5*time DO OD
    OD
    IF key=7 THEN
        key=255
        time==-32
    FI
    Sound(0,L*4+20,10,8)
    PA=Locate(x+A,y)
    PB=Locate(x,y+B)
    FOR ctr=0 to 100 DO OD
    SndRst()
    L=0
    IF PA)2 AND PB)2 THEN
        A=-A
        B=-B
    ELSEIF PA)2 AND PB(3 THEN
        A=-A
        color=2
        Plot(x,y)
        y=y+B
        color=9
        Plot(x,y)
    ELSEIF PB)2 AND PA(3 THEN
        B=-B
        color=2
        Plot(x,y)
        x=x+A
        color=9
        Plot(x,y)
    ELSEIF fate)127 THEN
        B=-B
    ELSE
        A=-A
    FI
    kolor==+1
    IF kolor)14 THEN
        kolor=1
    FI
    attract=0
OD
RETURN

PROC draw()
  BYTE qq

drawscreen()
x=40
y=95
DO
    IF key=28 THEN
        key=255
        drawscreen()
    ELSEIF key=33 THEN
        key=255
        bounce()
    FI
    IF Stick(0)=15 THEN
        flash()
    ELSEIF Stick(0)=7 THEN
        x=x+1
    ELSEIF Stick(0)=6 THEN
        x=x+1
        y=y-1
    ELSEIF Stick(0)=14 THEN
        y=y-1
    ELSEIF Stick(0)=5 THEN
        x=x+1
        y=y+1
    ELSEIF Stick(0)=11 THEN
        x=x-1
    ELSEIF Stick(0)=10 THEN
        x=x-1
        y=y-1
    ELSEIF Stick(0)=13 THEN
        y=y+1
    ELSEIF Stick(0)=9 THEN
        x=x-1
        y=y+1
    FI
    wallchex()
    IF Strig(0)=0 THEN
        color=0
        flash()
    ELSE
        color=15
    FI
    Plot(x,y)
    IF Stick(0)<>15 THEN
        qq=Strig(0)
        Sound(0,(200-x-y)*qq,8+2*qq,4)
        FOR ctr=0 to 1000 DO OD
        SndRst()
    FI
OD
RETURN
```

•

(continued from page 24)

Line 4550 — Compute ASCII code of distance clue, and (4560) jump ahead (to 4700) to display that clue.

Line 4600 — Find horizontal and vertical directions of treasure from Seeker.

Line 4610 — Compute the proper index num-ber for the ARROW( ) array.

Line 4620 — Make clue that arrow.

Line 4700 — Show the clue on the screen.

Line 4710 — Return to game loop.

That's the whole clue-making process. The computation of distance or of the proper arrow index may seem complex, but after puzzling them out, they begin to make sense.

Endings.

When the game ends, it branches to Line 5000 for the "End" routine. The elapsed time is computed using two of the time locations, and then is printed out, along with the number of guesses the player took.

If elapsed time was less than fifteen seconds, a little congratulatory sequence occurs on Line 5130. Lines 5140 to 5200 handle the option of playing again. The PLAYAGAIN variable is set to "one" if the START key is hit; if anything else is hit, it is set to "zero."

Only the beginning.

Why am I rehashing old game ideas (you may ask yourself)? Well, it so happens that this particular game idea is ideal to program simply and to expand upon creatively. With it, we can start small and think big.

For example, CLUES.A is only a one-player game. What happens when you make it two-player? I had a few ideas along those lines the other day, and I wrote them down in the following cryptic form:

Permutations of "Clues"
(two-player)

1. One treasure, *misinformation* beyond a certain range (say, five grid points).

2. Two treasures, *mixed information*. Or three treasures (1 or 2 treasures could be false)!

3. *Separate treasures* for each player: a. 1 /player/, *misinfo*; or b. 2/player, *mixed info*. Players' clues are distinguishable by color or RVS field.

4. Clue characteristic: a. *Transient* — clue disappears when you move away; or b. *Permanent* — clue remains visible (like in CLUES.A).

5. Special features: a. *Hidden Bomb*, if set off, moves treasure to a new spot; or b. *Mobile treasures* — treasures move after each guess (clues should be transient).

6. Weird/idea: Players take turns at screen. On each turn, a player sees only his treasure (which he must keep hidden) and himself. He may either: a. Take a guess as to where the opponent's treasure is; or b. Move his own treasure.

These notes may seem a bit mangled, but there are some interesting ideas in there. Of course, we don't have to develop all these possibilities at once. We can write various prototypes to try out different ideas. As a matter of fact, that's the subject of the next **Our Game**. Keep your booties on and stay tuned!

I want mail.

I want mail so badly I can taste it (no, that's just an expression; I don't eat the letters you send me). More importantly, I want YOU to vote in *Our Game Special Election-Year Game Idea Vote*! Remember, if you don't vote soon, Victor the Frightening Vote-Counting Robot will get angry — and you wouldn't want that to happen, would you? For details, take a look at last month's **ANALOG** (issue 19).

Of course, if there's anything you want to flame about, or any game idea you think is up to scratch, send it along, too. I promise you I'll read your letter.

Send your letters (and your favorite recipe for onion dip) to:

Our Game
c/o ANALOG Computing
P.O. Box 23
Worcester, MA 01603

Next month: more CLUES! □

```
100 REM - CLUES Prototype A
110 REM - by Joel Gluck / April '84
120 REM - "Our Game"
130 REM - ANALOG Computing June '84
200 GOSUB 1000:REM - Intro/Options
210 GOSUB 2000:REM - Initialize
220 GOSUB 3000:REM - Init. Screen
230 GOSUB 4000:REM - Game
240 GOSUB 5000:REM - End
250 IF PLAYAGAIN=1 THEN 220
260 END
1000 REM - Intro/Options
1100 GRAPHICS 0
1110 ? "Welcome to CLUES (version A)!"
1112 ? :? "     'A minute to learn,":?
 "     two minutes to master.'"
1120 ? :? "Object: Find the hidden tre
asure":? "as fast as you can."
1130 ? :? "Directions: Use joystick #1
 to move":? "your Seeker (the *) to a
point on the"
1140 ? "grid where you think the treas
ure":? "might be and then press the jo
ystick"
1150 ? "trigger. If you are correct, y
ou win! If not, a CLUE will appear.":?
 "The CLUE will be either an arrow or"
1160 ? "a number. An arrow points in t
he":? "general direction of the treasu
re."
1170 ? "A number tells your approximat
e":? "distance from the treasure."
1180 ? :? "Good luck and have fun!"
1190 ? :? "Press START to begin...";
1200 IF PEEK(53279)<>6 THEN 1200
```

```
1210 SETCOLOR 4,12,6:REM - acknowledge
 key press by changing border color
1220 RETURN
2000 REM - Initialize
2100 DIM XS(15),YS(15):REM - joystick
direction storage
2200 FOR Z=5 TO 15:REM - possible
joystick values
2210 READ A,B:REM - READ direction
values into dummy variables
2220 XS(Z)=A:YS(Z)=B:REM - store
direction values
2230 NEXT Z
2240 REM - direction values:
2250 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,-1,
-1,0,0,0,0,1,0,-1,0,0
2300 REM - ARROW graphics values:
2310 DIM CH$(1),ARROW(8):REM - arrow g
raphic storage
2320 FOR Z=0 TO 8:REM - possible arrow
 values
2330 READ CH$:ARROW(Z)=ASC(CH$(1,1))+1
28:REM - read arrow char. and convert
to  numerical code +128 for reverse
2340 NEXT Z
2350 REM - arrow characters:
2360 DATA ┌,<,└,^,?,V,┐,>,┘
2400 REM - other graphics:
2410 WALL=ASC("■")
2420 GRID=ASC(".")
2430 SEEKER=ASC("●")
2440 NUMBER=ASC("0")+128:REM - base
number is zero, +128 for reverse
field
2500 RETURN
3000 REM - Init. Screen
3100 GRAPHICS 0
3110 SETCOLOR 2,0,0:SETCOLOR 4,7,4:REM
 - background and border color
3120 POKE 752,1:REM - make cursor
invisible
3200 COLOR WALL
3210 PLOT 0,0:DRAWTO 39,0:DRAWTO 39,23
:DRAWTO 0,23:DRAWTO 0,0:REM - draw
outside wall
3250 COLOR GRID
3260 FOR X=1 TO 38:REM - grid breaks
down into 38 columns
3270 PLOT X,1:DRAWTO X,22:REM - draw
one column of grid pattern at X
3280 NEXT X
3300 XSK=19:YSK=11:REM - starting
coordinates of the Seeker
3310 COLOR SEEKER:PLOT XSK,YSK:REM -
draw Seeker
3320 UNDER=GRID:REM - "underneath" the
 Seeker is blank grid space
3400 XTRS=INT(RND(1)*38)+1:REM - x-
coordinate of the treasure
3410 YTRS=INT(RND(1)*22)+1:REM - y-
coordinate of the treasure
3420 IF XTRS=XSK AND YTRS=YSK THEN 340
0:REM - prevent treasure from being
right under Seeker's feet at start!
3500 RETURN
4000 REM - Game
4100 POKE 20,0:POKE 19,0:REM - set
timer to zero
4110 GUESSES=0
4120 FOR V=15 TO 0 STEP -0.5:SOUND 0,1
00,10,V:NEXT V:REM - starting bell
4200 REM - game shell
4210 S=STICK(0):T=STRIG(0):REM - get
stick and trigger values
4220 IF T=0 AND S=15 THEN GOSUB 4500:G
OTO 4210:REM - take a guess (trigger
was hit)
4230 IF S<>15 THEN GOSUB 4300:GOTO 421
0:REM - move Seeker
4240 GOTO 4210:REM - nothing happened
4300 REM - move Seeker
4305 POKE 77,0:REM - prevent ATTRACT
MODE
4310 XD=XS(S):YD=YS(S):REM - direction
 based on joystick value
4320 LOCATE XSK+XD,YSK+YD,G:REM - find
 out what's ahead of the Seeker
4330 IF G=WALL THEN RETURN :REM - can
not move through a wall!
4340 SOUND 0,XSK+YSK,8,4
4350 COLOR UNDER:PLOT XSK,YSK:REM -
erase Seeker
4360 XSK=XSK+XD:YSK=YSK+YD:REM - move
Seeker
4370 COLOR SEEKER:PLOT XSK,YSK:REM -
draw Seeker
4380 UNDER=G:REM - new "underneath"
value
4390 SOUND 0,0,0,0
4400 RETURN
4500 REM - take a guess
4502 GUESSES=GUESSES+1
4505 IF XSK=XTRS AND YSK=YTRS THEN POP
 :RETURN :REM - win!
4510 FOR V=8 TO 0 STEP -0.5:SOUND 0,10
,4,V:SETCOLOR 2,0,V:NEXT V:REM -
 special effects
4520 IF UNDER<>GRID THEN 4700:REM -
show the same old clue that's in that
spot
4530 DIST=INT(SQR((XTRS-XSK)^2+(YTRS-Y
SK)^2)+0.5)
4540 IF DIST>9 OR RND(1)<0.3 THEN 4600
:REM - if too far, use an arrow clue
4550 UNDER=DIST+NUMBER:REM - number
clue
4560 GOTO 4700
4600 XD=SGN(XTRS-XSK):YD=SGN(YTRS-YSK)
:REM - direction toward treasure  from
 Seeker
4610 Z=3*(XD+1)+(YD+1):REM - compute
proper arrow number
4620 UNDER=ARROW(Z):REM - use that
arrow
4700 SOUND 0,20,10,8:COLOR UNDER:PLOT
XSK,YSK:SOUND 0,0,0,0:REM - show clue
4710 RETURN
5000 REM - End
5100 TIME=(PEEK(20)+256*PEEK(19))/60:R
EM - find jiffies and divide by 60 to
get seconds.
5110 GRAPHICS 0
5120 ? :? "You did it in ";TIME;" seco
nds"
5125 ? "(and in ";GUESSES;" guesses)."
5130 IF TIME<15 THEN ? :FOR F=100 TO 1
0 STEP -10:SOUND 0,F,10,4:? "Congratul
ations!":NEXT F:SOUND 0,0,0,0
5140 ? :? "Hit START to play again,"
5150 ? "anything else to stop...";
5160 POKE 764,255:REM - clear keyboard
 value
5170 IF PEEK(764)<>255 OR PEEK(53279)<
6 THEN PLAYAGAIN=0:? :POKE 764,255:RET
URN :REM - they've had enough
5180 IF PEEK(53279)=6 THEN PLAYAGAIN=1
:RETURN :REM - more!
5200 GOTO 5170
```

●

CHECKSUM DATA

(See page 23)

```
100 DATA 244,235,824,330,529,922,868,5
98,273,700,46,202,45,918,124,6858
1120 DATA 408,447,457,107,985,441,461,
135,606,407,787,905,98,178,471,6893
2220 DATA 389,546,19,800,336,280,500,9
70,550,838,925,160,479,253,441,7486
2440 DATA 19,796,550,49,388,142,115,32
2,121,446,769,547,23,344,778,5409
3400 DATA 838,822,418,798,56,979,900,7
33,345,583,196,745,955,683,896,9947
4310 DATA 344,925,529,199,196,763,352,
875,222,797,679,377,345,267,67,6937
4530 DATA 790,670,331,739,441,962,556,
276,807,671,167,54,396,528,893,8281
5140 DATA 3,720,944,867,845,737,4116
```

●

MOVIEMAKER
by Interactive Picture Systems
RESTON SOFTWARE
11480 Sunset Hills Road
Reston, VA 22090
(800)336-0338
48K Disk $49.95

by Bob Curtin

Here's another good one, folks. **MovieMaker** is a program or, more correctly, a series of programs which allow the user to create animated "movies" about thirty to sixty seconds long, with a very professional appearance. The animated computer graphics which can be created with this package are as good as any the "big boys" make, and you're limited only by time and your imagination.

MovieMaker is broken up into four sub-programs entitled compose, record, smooth, and play. The compose mode allows you to draw characters (the actors) as a series of shapes which, when combined into sequences, give the illusion of movement. Each shape is the equivalent of a single drawing in regular animation, and up to sixteen shapes can be combined to create an action sequence. This would be the equivalent of flipping sixteen pages or "frames" in normal animation. Many action sequences can be strung together and/or played simultaneously, to create a full-length feature up to 300 frames long.

Drawing shapes in this mode is much the same as in Datasoft's **MicroPainter** or Atari's **Paint** program — that is, with the joystick and keyboard. However, several clever little features take a lot of the work out of drawing scores of tiny, repetitious shapes. The program allows you to duplicate and move shapes around. These shapes can then be altered and duplicated, and altered and duplicated again, and so on [you get the picture? (pun intended)]. A mirror-image feature is provided, so you only have to draw half of symmetrical shapes; the computer will draw in the other half automatically. As in the other graphics generation programs, there's a zoom feature that makes attending to details in your drawings a lot easier. The zoom feature can also be used when recording your movie, to give the effect of a shape coming closer or vice versa.

Most of the work is done in the compose mode, creating the action sequences and the backgrounds. Multiple shape files can be stored on disk for use later, when you start the actual "filming" of your movie in the record mode. These shape files can also be used to build backgrounds, and accumulated for use in sequences in future animations.

Once the shapes, sequences, and backgrounds are completed, these elements can be combined in the record mode, to create the finished film. There are a variety of controls in this mode, which allow you to record up to six "actors" on the screen at the same time. This is accomplished by re-recording additional sequences over an existing film. You can start and stop anywhere in the film, adding actors, changing colors, zooming, changing the recording speed (similar to filming in fast or slow motion), adding sound and fine tuning the whole creation by using the many editing commands.

Once the recording is done and the animation saved to disk, the next step is the easiest. Enter the smooth mode and give the computer some room. Your Atari will go over the film and take out all of the silent-film jumpiness — giving you back a work of art.

MovieMaker is a complex package, and with complexity comes a learning curve. There's a lot to remember. There's a myriad of controls, procedures and commands, and it takes a while to learn them all. Once learned, however, the complexity will be appreciated for the control that it gives you over the animated graphics you can create with this program.

The documentation provided with **MovieMaker** is in the form of a hundred-odd page booklet with step-by-step instructions, tips for advanced users, a trouble-shooting section, glossary, detailed summary of commands and (lo and behold) an accurate and detailed table of contents. The booklet is well written, clean, and (although it leaves nothing to chance) you don't get the impression that it was written for the crayon and bubblegum set. There are also several sample movie files provided, as an example of what this package is capable of when in the right hands.

Unfortunately, there's no way to play back these films, except with the **MovieMaker** program. Reston did not see fit to provide a subroutine or reproducible program which you could use to play back the animations you create. That's a gripe I had with both the Atari **Paint** program and Datasoft's **MicroPainter** program. It'd be a simple matter to list such a subroutine in the documentation, or provide one on the disk, that could be reproduced and used in your own programs. I grant that it would be more difficult with **MovieMaker**, but certainly within the realm of the possible. With software running nearly 20% of the cost of the computer it's used on, I think the software houses ought to pay at least as much attention to their customers as they do the software pirates. But that's another story.

It is a good buy. As I've said in the past, these wondrous devices are capable of so much more than creating hordes of killer tomatoes. It's both refreshing and gratifying to see programs as good as this on the market. When my kids put aside their game disks and booted up **MovieMaker**, I was interested to see their reaction. Well, several weeks later they're still at it — struggling at times, but sticking to it and obviously enjoying themselves. I think Reston has a winner. □

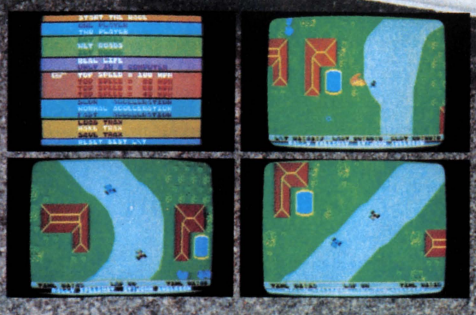
John Anderson's

RALLY SPEEDWAY

SLIP INTO SOMETHING TURBO-CHARGED

like the road-scorching speedster in RALLY SPEEDWAY — the game that turns your ATARI into a full-blown, four-wheeled demon! Take on our demanding course, or use the unique Construct-A-Track option to design your own. Push yourself to the limit for a better lap time, or get a friend in on the action for a one-on-one duel to the finish line — it's your choice!

CALL TOLL FREE 1-800-327-7172



MAKE TRACKS TO YOUR DEALER TODAY!

ATARI 400/800/1200 CARTRIDGE

053-0171 . . . . . $49.95

If your dealer doesn't have Rally Speedway in stock, phone 1-800-327-7172 with your order, or write us at Adventure International, P.O. Box 3435, Longwood, FL 32750 for your FREE catalog.

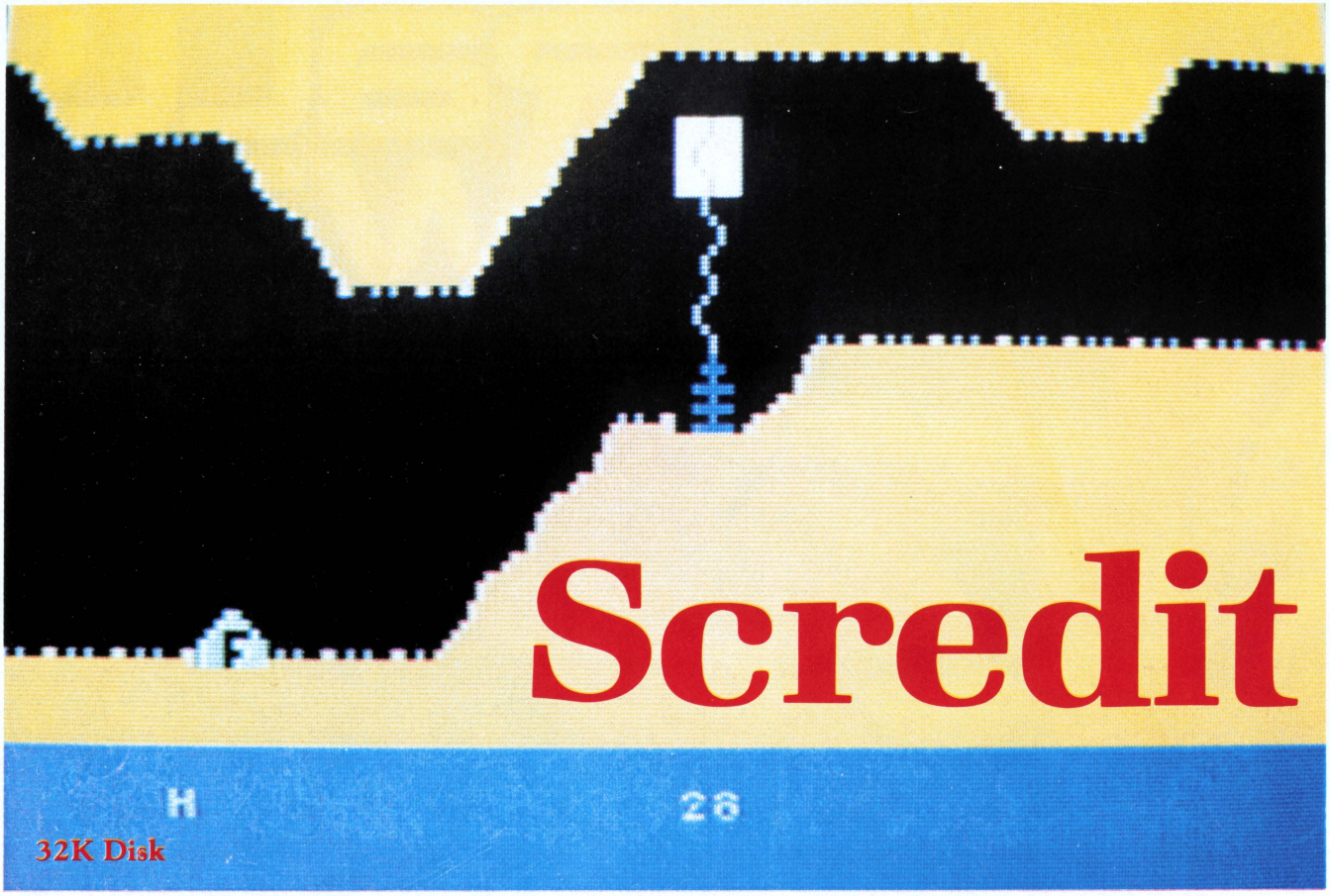
DEALER ORDERS WELCOME

CIRCLE #150 ON READER SERVICE CARD.

155

Adventure® INTERNATIONAL

A DIVISION OF SCOTT ADAMS, INC.



by Ron Hodge

Smooth scrolling is the most dramatic graphics capability of the Atari system. If you've been following Kyle Peacock's recent series, **Fine Scrolling** (issues 13-16), you've no doubt sat entranced while the screen moved about smoothly under joystick control.

After deciding to set up your own scrolling game or utility, you whip out **Create-A-Font** (issue 16) and create the character set that will make your screen sparkle. Then, it suddenly dawns on you — how do you design a scrolling screen when you can't even see all of it?

I was faced with the same problem, and the result was **Scredit** (short for Scrolling Screen Editor). **Scredit** is a screen editor that allows you to design a screen simply by typing in the characters to the screen, then SAVEing the screen as a binary file.

My particular need was for a screen in Graphics Mode 2, 24 TV screens wide. While your screen may not be the same size or mode, I think you'll find it easy to modify the program for your own needs. Let's take a look at how **Scredit** works.

Scredit starts with a menu of options you'll need to get started. After taking a look at the disk directory with option 4, press 1 or 3 to LOAD in a screen or character set. The menu will remind you whether you're LOADing or SAVEing a file, and ask for the filename. Prefix the filename with D:. **Scredit** will confirm that the LOAD is complete, or report any disk I/O errors, using standard error numbers.

If you change your mind about LOADing or SAVEing a file, simply press RETURN when asked for the filename. You'll get an error message and a request for your next option. (You may LOAD or SAVE a file from any drive, but only look at the directory on the drive 1.)

After LOADing your files, press 5 to exit to the editor. Your cursor will be in the upper left corner of the screen. To move it around, use the CONTROL-ARROW keys or press the trigger while moving the joystick. If you want to place a character on screen, press the appropriate key, or the space bar to erase a character. All CONTROL-KEY combinations other than the cursor arrows are ignored by **Scredit.**

Press OPTION to change the image priority of the cursor. You have two methods to see which character is under the cursor. Pressing START will toggle the screen between the character set chosen and the internal character set. Pressing SELECT will display the character in the text window, along with its hex value.

The hex value will show you how the character derived its color. If the high bit is set, it shows an inverse character. If the next-highest bit is set, the character is in lower caps. For example:

C=$23	upper case
C=$63	small caps (+$40)
C=$A3	inverse (+$80)
C=$E3	inv + small caps (+$40+$80)

Move the joystick to scroll the screen. The cursor will move if the trigger is pressed. You can get back to the menu by pressing ESCAPE. Flipping to the menu and back is also convenient for positioning the cursor at the beginning of the screen, since you always enter the editor at the beginning of the screen.

That's all there is to using **Scredit.** It shouldn't be hard to modify **Scredit** for other screen sizes or modes. Screen and player/missile parameters, in addition to the display list, are all the changes needed. The logic flow remains the same. Since horizontal and vertical scrolling are so similar, the routines to maintain cursor position for vertical or horizontal scrolling are similar to the current routines.

One last thought. Since the logic for **Scredit** is so minimal — about 2300 bytes — it could be coresident with a program under development. Then you could flip between the program you're fine-tuning and **Scredit** to make screen changes as needed. □

BASIC Listing.

```
10 REM *** SCREDIT ***
20 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
0,0,10,11,12,13,14,15
30 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 110:? "CHECKING DATA"
40 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 160
50 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
60 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
70 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 80
80 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
90 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
THEN 40
100 GOTO 160
110 IF PEEK(195)<>6 THEN 160
120 IF PASS=2 THEN PUT #1,224:PUT #1,2
:PUT #1,225:PUT #1,2:PUT #1,0:PUT #1,6
4:CLOSE #1:END
130 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
140 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,64:PUT #1,80:PUT #1,73
150 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 110:GOTO 40
160 ? "BAD DATA: LINE ";LINE:END
1000 DATA 202843202F40207940201C4320A3
40208DC40207D41A580C93590EED0062028434C
F144202843203C432040434C,627
1010 DATA 0040A220A9039D4203A94D9D4403
A9439D4503A9089D4A03A9009D4B032056E4A2
10A9039D4203A94A9D4403A9,599
1020 DATA 439D4503A90C9D4A03A9009D4B03
2056E4A9008DC502A99A8DC6028DC80260A230
A9039D4203A9509D4403A943,993
1030 DATA 9D4503A9049D4A03A9009D4B0320
56E478A510297F85108D0ED25860A9008DA302
A9408DA402A20FBCB444BDB3,542
1040 DATA 44868220C140A682CACA10EF60A2
009D4403989D4503A9099D4203A9289D48038A
9D49032056E46020F542A907,162
1050 DATA A230A0012063341A99B8DCA44A9C9
A04420C140201443ADC944C920F0DEC9319005
C937B00160A9FD8DC944A9C9,737
1060 DATA A04420C14020F542A9208DC9444C
E840A240A9039D4203A9069D4A03A9539D4403
A9439D45032056E4A905A240,304
1070 DATA A0282063419810078481A2404CEC
42A220A9C49D4403A9449D4503A9099D4203A9
282071414C37419D4203A9C9,608
1080 DATA 9D4403A9449D4503989D4803A900
9D49032056E460ADC9448580C9349009D00620
1E414C024360C932D00AA9FD,112
1090 DATA A04320C1404CA441A916A04420C1
4020143A9098555202243A905A210A00F2063
41201C43981007C088F0034C,961
1100 DATA 5442ADC944C944F005A0A54C5442
A580C933D033A24020EC42A0042075429810034
4C5442A907A2409D4203A900,555
1110 DATA 9D4403A9549D4503A9009D4803A9
029D490320AE42A4814C3442A24020EC42A580
C931D004A004D002A0082075,835
1120 DATA 429810034C5442A580C931D004A9
07D002A90B2095428481981003 4C5442A580C9
32D00AA940A04420C1404C02,114
1130 DATA 43A92FA04420C1404C0243A202A9
209D7544CA10FA20B742A99BA00C996A442014
43A96AA04420C1404C0243A2,556
1140 DATA 40A9039D4203A9C99D4403A9449D
4503989D4A03A9009D4B032056E4848160A240
9D4203A9009D4403A9609D45,915
1150 DATA 03A9009D4803A9149D49032056E4
848120EC42608481 98A009A200C9649006E964
E84CBE4220E242A200C90A90,711
1160 DATA 06E90AE84CCD4220E24218693099
6A44C860488A6930996A44C86860A90C9D4203
2056E460A9188555A9008556,401
1170 DATA A90C85546020144 3A951A04420C1
40A906CD1FD0D0FB60A991A04420C14060A201
8EF00260A2008EF00260A210,345
1180 DATA 20EC42A22020EC42A23020EC42A2
404CEC426C0A0060A97AA04420C1404C024345
3A9B533A9B4B3A9B443A2A2E,602
1190 DATA 2A007D1D1D7F1E1E1E1E53435245
4449542028432920313938342052 4F4E20484F
4447459B1D1D7F312D204C4F,309
1200 DATA 414420534352454E9B7F322D20
534156452053435245454E9B7F332D204C4F41
44204348415220534554 9B7F,680
1210 DATA 342D2044313A2044495245435 44F
52599B7F352D2045584954 20544F2045444954
4F529B7F362D204558495420,876
1220 DATA 544F20444F539B1D7F5748494348
204F5054494F4E203F20202020202020209B1D7F
4E414D45204F462046494C45,908
1230 DATA 20544F20D3C1D6C5203F9B1D7F4E
414D45204F462046494C4520544F20CCCFC1C4
203F9B1D7F4C4F414420434F,45
1240 DATA 4D504C455445449B1D7F53415645
20434F4D504C45545449B7F505245535320D3
D4C1D2D420544F20434F4E54,167
1250 DATA 494E55459B7FC5D2D2CFD23A2020
2020202020209B7FC5D2D2CFD23A204E4F2044
4F53204F4E204449534B9B20,634
1260 DATA 2020202020202020202020202020
2020202020202020202020202020202020202020
9BE543D543C243B0439E438E,292
1270 DATA 437C435943202020200000000000
0000000030000000000000000000000000000000
0000000000000000000000,866
1280 DATA 0000000000A00FB9AE4899F00088
10F7A004B9A94899C4028810F7A9548DF402A9
00A0FF99003488D0FA85F08D,141
1290 DATA 04D48583A95A8DC002A9308D00D0
85FDA9008D01D08D02D08D03D0A9018D6F02A9
008D08D0A9FF207F47A9308D,351
1300 DATA 07D4A9038D1DD0A0A1A90099FF73
88D0FA78A9008D2F02A02FB979489900068810
F7A9008D3002A9068D3102A9,245
1310 DATA 3E8D2F02A9478D0002A9488D0102
58A096A245A907205CE4A9C08D0ED42079404C
B545A583D018A6FEF00CA514,78
1320 DATA 2901D00E20B6474C62E4AD8402F0
03205847 4C62E4A514C514F0FCA5FED0F6AD84
02D02BAD7802C907D0062087,381
1330 DATA 464C4746C90BD00620BD464C4746
C90ED00620ED464C4746C90DD006201E474C47
46AD1FD0C903D0062018484C,399
1340 DATA 1046C905D009206C482029484C10
46C906D00B208E48A9048D1FD04C4746ADFC02
C9FFF01DC91CD0034C0C4920,702
```

```
1350 DATA CF48B01120E848B00F205A46206D
46207E464C47464CB545A9FF8DFC024C1046A9
FF8DFC02A204A514C514F0FC,735
1360 DATA CAD0F74CB545A230A9009D48039D
4903A9079D42032056E460A82A2A2A2A2903AA
98299F1D284985F160205748,111
1370 DATA A000A5F191FAA4F8C013F015B013
E6F8E6F2D002E6F3A5FD18690885FD8D00D060
A90085F8A93085FD8D00D0A5,837
1380 DATA F238E91385F2B006A5F3E90085F3
60A4F8F015C6F8A5F2D002C6F3C6F2A5FD38E9
0885FD8D00D060A91385F818,813
1390 DATA 65F285F2A5F3690085F3A9C885FD
8D00D060A4F9F013C6F9C6F3C6F3A900207F47
84FCA9FF207F4760A90985F9,714
1400 DATA A5F318691285F3A900207F47A08F
84FCA9FF207F4760A4F9C009F01AB018E6F9E6
F3E6F3A900207F47A5FC1869,979
1410 DATA 1085FCA9FF207F4760A90085F9A5
F338E91285F3A900207F47A02F84FCA9FF207F
4760AD7802C907D00E208A47,478
1420 DATA B01AA90885F0A98085FE60C90BD0
0D209E47B008A90085F0A91485FE60A4FCA210
91F688CAD0FA60A4F4C0E190,948
1430 DATA 04A5F5D008E6F4D002E6F5186038
60A4F4C0069004C6F41860A6F5F008C6F41002
C6F518603860 8A102BC6F0F0,34
1440 DATA 06A5F08D04D460A21CFE0306D003
FE0406CACACA10F3A90085FEE6F2D002E6F3A9
0885F08D04D460E6F0A4F0C0,471
1450 DATA 08B006A5F08D04D460A21CBD0306
D003DE0406DE0306CACACA10F0A90085FEA5F2
D002C6F3C6F2A90085F08D04,184
1460 DATA D460A901CD6F02D006A9048D6F02
608D6F0260A000B1FA484A4A4A4A2038486829
0F186910C91A900318690799,150
1470 DATA 3C74C86048A9E08D0AD48D09D4A5
FF8DC6026840A5F285FAA5F385FBA5F0F008A5
FAD002C6FBC6FA6020574 8A0,945
1480 DATA 00B1FA293F8D2D74607070705700
605700625700645700665700685700 6A57006C
57006E570070D70072420074,440
1490 DATA 42287442507442787441000628CA
94460008FF026005000034000000002F300094
A954CDF402F0048DF40260A9,18
1500 DATA E08DF4026048293FA005D9E248F0
068810F868186068386 00C1C2C343736C98FD0
05201E473860C98ED00520ED,619
1510 DATA 463860C986D00520BD463860C987
D0042087463860A9FF8583A23020EC42A9008D
1DD08D00D0A9228D2F028D00,431
1520 DATA D44C00404000206000000000000
0000000000000000000000000000000000000000
00000000000000000000000,975
```

•

CHECKSUM DATA

(See page 23)

```
10 DATA 886,957,808,431,727,198,599,55
3,272,701,611,124,947,676,36,8526
160 DATA 165,400,643,736,739,594,825,6
10,609,552,355,530,502,378,633,8271
1140 DATA 619,561,666,493,760,573,382,
415,551,888,586,695,839,484,647,9159
1290 DATA 812,789,983,768,887,783,939,
678,46,956,72,145,914,964,983,10719
1440 DATA 982,990,696,217,152,410,681,
680,498,5306
```

•

(Assembly language starts on page 96)

Some program listings reproduced in **A.N.A.L.O.G.** may contain "strange" characters not shown on the ATARI keyboard. These are special characters which use the CTRL, ESC and "ATARI LOGO" (INVERSE) keys. Shown below is a list of these characters and the keystrokes used to get them. □

♥ --- CTRL ,
├ --- CTRL A
┤ --- CTRL B
┘ --- CTRL C
┴ --- CTRL D
┐ --- CTRL E
╱ --- CTRL F
╲ --- CTRL G
◢ --- CTRL H
▪ --- CTRL I
◣ --- CTRL J
▝ --- CTRL K
▘ --- CTRL L
▔ --- CTRL M
▁ --- CTRL N
▗ --- CTRL O
♣ --- CTRL P
▖ --- CTRL Q
─ --- CTRL R
┼ --- CTRL S
● --- CTRL T
▄ --- CTRL U
▏ --- CTRL V
┬ --- CTRL W
┴ --- CTRL X
▌ --- CTRL Y
└ --- CTRL Z
ESC ESC
↑ --- ESC CTRL UP-ARROW
↓ --- ESC CTRL DOWN-ARROW
← --- ESC CTRL LEFT-ARROW
→ --- ESC CTRL RIGHT-ARROW
● --- CTRL .
♠ --- CTRL ;
ESC SHIFT CLEAR
ESC BACK S
ESC TAB
INVERSE CTRL ,
INVERSE CTRL A
INVERSE CTRL B
INVERSE CTRL C
INVERSE CTRL D
INVERSE CTRL E
INVERSE CTRL F
INVERSE CTRL G
INVERSE CTRL H
INVERSE CTRL I
INVERSE CTRL J
INVERSE CTRL K
INVERSE CTRL L
INVERSE CTRL M
INVERSE CTRL N
INVERSE CTRL O
INVERSE CTRL P
INVERSE CTRL Q
INVERSE CTRL R
INVERSE CTRL S
INVERSE CTRL T
INVERSE CTRL U
INVERSE CTRL V
INVERSE CTRL W
INVERSE CTRL X
INVERSE CTRL Y
INVERSE CTRL Z
ESC DELETE
ESC INSERT
ESC CTRL TAB (CLR)
ESC SHIFT TAB (SET)
INVERSE SPACE
INVERSE _
INVERSE CTRL .
INVERSE CTRL ;
INVERSE |
ESC CTRL 2
ESC CTRL BACK S
ESC CTRL INSERT
```
```

Get Serious, Go Ape With An

APE • FACE™

Parallel Printer Interface For Atari® Computers

Model XLP
Atari 400/800
Atari 600XL/800XL

Model 12XLP
Atari 1200XL

$89.95 *At the dealer you trust most.*

Ape•Face makes it easy to expand your Atari Computer—so you can do *more* than play games! Choose any Centronics standard parallel printer to enhance your system, like Epson, Gemini, or C. Itoh.

Ape• Faces are complete with cables and are friendly with all Atari hardware and software. Easy connection through the serial peripheral port makes installation a snap.

Only APE tested quality products receive the Stomp of Approval.

STOMP OF APPROVAL

A Product of
DIGITAL DEVICES Corporation ™

430 Tenth Street
Suite N 205
Atlanta, GA 30318
(404) 872-4430

Assembly language listing.

```
; SCREDIT (C) 1984 RON HODGE
;
; ASSEMBLED ON MAC65
;
        .OPT NO LIST
        .OPT OBJ
;
;-------------------------------
; PROGRAM EQUATES
;-------------------------------
DLIST   =   $0600
PLBAS   =   $3000   PM BASE
PB0     =   $3400   PLYR 0 BASE
HXTXT   =   $743C   CHR UNDER CRSR
CHDIS   =   $742D
SCREEN  =   $6000   1ST SCRN BYTE
SCRSIZ  =   3120    SCRN SIZE
CHRSET  =   $5400   CHR SET LOC
CHRSIZ  =   512     CHR SET SIZE
;
;-------------------------------
; ZERO PAGE EQUATES
;-------------------------------
;
OPT     =   $80     MENU OPTION
STAT    =   $81     I/O ERR STATUS
XSAV    =   $82     TEMP X VALUE
MODFLG  =   $83     MENU/EDIT STATUS
SCBIT   =   $F0     SCROL BIT CTR
CHR     =   $F1     DISP CHAR
SCRN    =   $F2     PTR TO SCRN
SCRNH   =   $F3
SCRLO   =   $F4     SCRL BYTE CTR
SCRHI   =   $F5
PMLO    =   $F6     PTR TO PM0
PMHI    =   $F7
COL     =   $F8     CURSOR COLUMN
ROW     =   $F9     ROW
TEMP0   =   $FA     TEMP POINTER
TEMP1   =   $FB
VOFF    =   $FC     VRT OFFSET P0
HORP0   =   $FD     HPOSP0 SHADOW
SCFLG   =   $FE     SCROL IF <>0
;                       POS = RITE SCROL
;                       NEG = LEFT SCROL
TXTCLR  =   $FF     TXT WINDOW COLR
;
;-------------------------------
; CONSTANTS
;-------------------------------
;
RITPM   =   200     RITEMOST P0 POS
LFTPM   =   48      LEFT    "
BOTPM   =   191     BOT     "
TOPPM   =   47      TOP
ROWM    =   9       BOTTOM ROW
COLM    =   19      RITE COLUMN
WIDTH   =   255     CURSOR WIDTH
HITE    =   16      HITE
START   =   6       CONSOL VALUES
SELECT  =   5
OPTION  =   3
HOR     =   8       HOR DISTANCE
RETURN  =   $0C     KEYS
ESC     =   $1C
TAB     =   $2C
DEL     =   $34
INSERT  =   $37
CLR     =   $36
COLON   =   $3A
SPACE   =   $20
EOF     =   $88     END OF FILE
EOL     =   $9B     END OF LINE
BUFLEN  =   40
EDEV    =   $10     DEVICE NUM'S
SDEV    =   $20
KDEV    =   $30
DDEV    =   $40
;
;-------------------------------
; SYSTEM EQUATES
;-------------------------------
;
CH      =   $02FC   KEY PRESSED
POKMSK  =   $10
IRQEN   =   $D20E
VDSLST  =   $0200
SIZEP0  =   $D008
COLPF2  =   $02C6
HSCROL  =   $D404
PMBASE  =   $D407
COLOR0  =   $02C4   COL REG 0
COLOR1  =   $02C5
COLOR4  =   $02C8
SDLSTL  =   $0230   PTR TO DLIST
SDLSTH  =   $0231
CHBAS   =   $02F4   CHAR SET
SDMCTL  =   $022F   SCRN CONTROL
DMACTL  =   $D400
GRACTL  =   $D01D   GRAPH  "
HPOSP0  =   $D000
GPRIOR  =   $026F
STICK0  =   $0278   J/STICK0
PCOLR0  =   $02C0
CONSOL  =   $D01F
XITVBV  =   $E462
TRIG0   =   $0284
CHBASE  =   $D409   CHAR SET ADR
WSYNC   =   $D40A
NMIEN   =   $D40E
ATACHR  =   $02FB
CRSINH  =   $02F0   CURS ON/OFF
COLCRS  =   $55     CURSOR COLUMN
ROWCRS  =   $54     ROW
TABMAP  =   $02A3   TAB SET POS'S
RAMTOP  =   106
;
;-------------------------------
; CIO ADDRESSES
;-------------------------------
;
ICCOM   =   $0342   COMMAND ADDR
ICBAL   =   $0344   BUFF ADDR LO
ICBAH   =   $0345   BUFF ADDR HI
ICAX1   =   $034A   AUX BYTE 1
ICAX2   =   $034B   AUX BYTE 2
CIOV    =   $E456   CENTRAL I/O
ICBLL   =   $0348   BUF LENGTH LO
ICBLH   =   $0349   BUF LENGTH HI
;
;-------------------------------
; CIOV COMMANDS
;-------------------------------
;
OPEN    =   3
CLOSE   =   12
GETCHR  =   7       GET CHARACTER
GETREC  =   5       GET RECORD
PUTCHR  =   11      PUT CHARACTER
PUTREC  =   9       PUT RECORD
OPDIR   =   6       OPEN DIRECTORY
WRITE   =   8       AUX COMMAND
READ    =   4       AUX COMMAND
;-------------------------------
;
        *=  $4000
;-------------------------------
; START OF SCREEN MENU ROUTINE
;-------------------------------
MENU
;
        JSR CLSALL  CLOSE DEVICES
        JSR MODE0   OPEN S: & E:
        JSR OPNKBD  OPEN KEYBOARD
A1
        JSR CRSROFF
        JSR SCRPRT  PRINT MENU
        JSR GETOPT  GET OPTION
        JSR DOOPT   DO THE OPTION
        LDA OPT
        CMP #'5     EXIT OPTION ?
        BCC A1      NOT IF <5
        BNE A2      IT'S 6
        JSR CLSALL  CLOSE DEVICES
        JMP INIT    GO TO EDITOR
A2
        JSR CLSALL
        JSR DOS
; RETURN ONLY IF NO DOS ON DISK
        JSR NODOS   ANNOUNCE NO DOS
        JMP MENU    THEN START OVER
;
;-------------------------------
; OPEN S:, E: AND K: DEVICES
;-------------------------------
;
MODE0
; OPEN SCRN TO MODE 0
        LDX #SDEV
        LDA #OPEN
        STA ICCOM,X
        LDA # <SCOLON
        STA ICBAL,X
        LDA # >SCOLON
        STA ICBAH,X
        LDA #8      AUX COMMAND
        STA ICAX1,X SET TO 'WRITE'
        LDA #0      GR. MODE
        STA ICAX2,X
        JSR CIOV
OPENED
; OPEN THE E:DEVICE
        LDX #EDEV
        LDA #OPEN
        STA ICCOM,X
        LDA # <ECOLON
        STA ICBAL,X
        LDA # >ECOLON
        STA ICBAH,X
        LDA #12     AUX COMMAND
        STA ICAX1,X READ & WRITE
        LDA #0
        STA ICAX2,X
        JSR CIOV
;
        LDA #0      CHANGE COLORS
        STA COLOR1
        LDA #154
        STA COLPF2
        STA COLOR4
        RTS
OPNKBD
        LDX #KDEV   OPEN KEYBOARD
        LDA #OPEN
        STA ICCOM,X
        LDA # <KCOLON
        STA ICBAL,X
        LDA # >KCOLON
        STA ICBAH,X
        LDA #4
        STA ICAX1,X
        LDA #0
        STA ICAX2,X
        JSR CIOV
        SEI         TURN OFF BRK KEY
        LDA POKMSK
        AND #$7F    STRIP HI BIT
        STA POKMSK
        STA IRQEN
        CLI
        RTS
;
;-------------------------------
; PRINT THE MENU SCREEN
;-------------------------------
SCRPRT
        LDA #0      CLR 8 TAB POS'S
        STA TABMAP
        LDA #$40    TAB AT 10
        STA TABMAP+1
PRTMSG
;
; THIS ROUTINE GETS THE MSG
; ADDR BEFORE JUMPING TO THE
; PRINT ROUTINE.
;
        LDX #15
B1
        LDY MSG,X   HI ADDR
        LDA MSG-1,X GET LO ADDR
        STX XSAV    SAVE X
        JSR PRINT
        LDX XSAV
        DEX
        DEX
        BPL B1
        RTS
PRINT
; ENTER WITH A=LSB OF MSG
;            Y=MSB OF MSG
        LDX #0
        STA ICBAL,X
        TYA
        STA ICBAH,X
        LDA #PUTREC
        STA ICCOM,X
        LDA #40     JUST TO BE SAFE
        STA ICBLL,X
        TXA
        STA ICBLH,X
        JSR CIOV
        RTS
;
;-------------------------
; GET THE OPTION FM K: DEVICE
;-------------------------
;
GETOPT
        JSR POS2412 POSITION CRSR
GETKEY
        LDA #GETCHR GET THE KEY
        LDX #KDEV
        LDY #1
        JSR DOCIO
PRT1
        LDA #$9B    FORCE EOL AS 2ND
        STA BUFF+1  CHAR IN BUFFER
        LDA # <BUFF SO ONLY 1 CHR
        LDY # >BUFF IS PRINTED.
        JSR PRINT   PRINT SELECTION
        JSR PRTBLK  SKIP A LINE
;
        LDA BUFF    IS KEY OK ?
        CMP #$20    IS IT SPACE ?
        BEQ GETOPT
        CMP #'1     KEY <1 ?
        BCC PRTQUE  YES,ASK KEY AGAIN
        CMP #'6+1   KEY >6 ?
        BCS PRTQUE
        RTS
PRTQUE
; PRINT QUESTION AGAIN AFTER
; WRONG KEY IS PRESSED
        LDA #$FD    BUZZER CHAR
        STA BUFF
        LDA # <BUFF
        LDY # >BUFF
        JSR PRINT
        JSR POS2412 PLACE CRSR
```

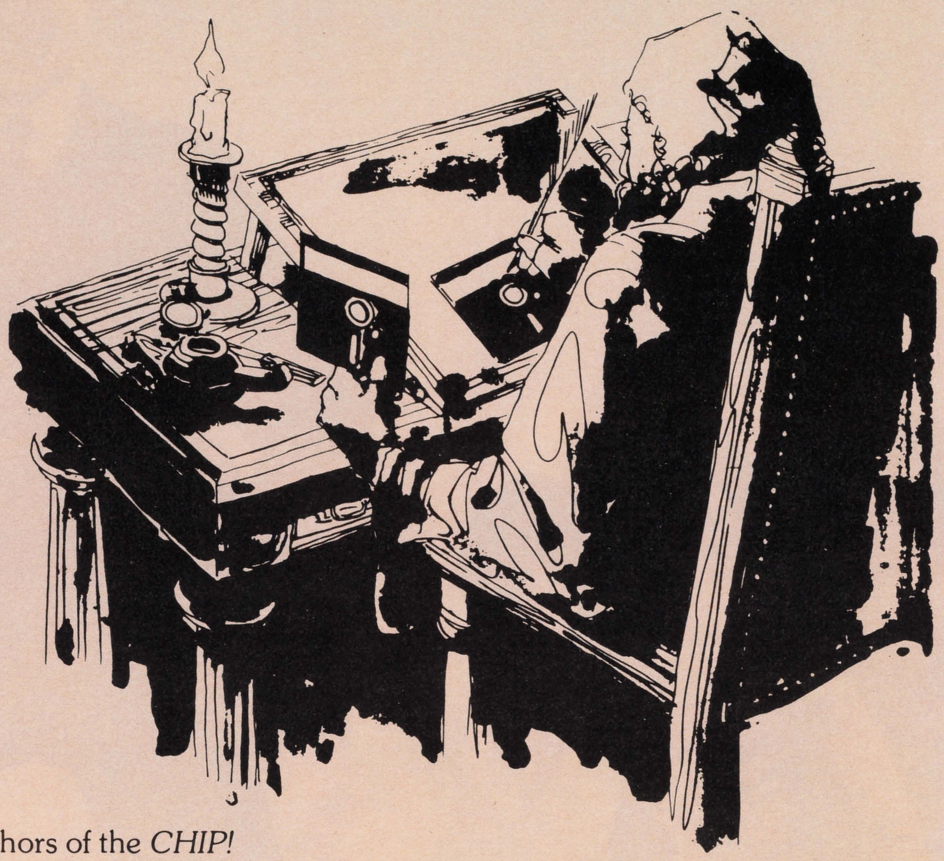
```
        LDA #$20    SPACE
        STA BUFF
        JMP PRT1    CLR AND CONTINUE
;
;-----------------------------
; PRINT THE DIRECTORY ON SCRN
;-----------------------------
GETDIR
        LDX #DDEV   OPEN DIRECTORY
        LDA #OPEN
        STA ICCOM,X
        LDA #OPDIR
        STA ICAX1,X
        LDA # <DIRNAM
        STA ICBAL,X
        LDA # >DIRNAM
        STA ICBAH,X
        JSR CIOV
GET1
        LDA #GETREC GET DIRECTORY
        LDX #DDEV
        LDY #40
        JSR DOCIO
        TYA
        BPL PRTDIR
        STY STAT    SAVE STATUS
        LDX #DDEV
        JMP DEVOFF
PRTDIR
; PRINT DIRECTORY ONE FILE
; AT A TIME
;
        LDX #SDEV
        LDA # <BUFF-5 FORCE 3 SPACES
        STA ICBAL,X IN FRONT
        LDA # >BUFF-5
        STA ICBAH,X
        LDA #PUTREC
        STA ICCOM,X
        LDA #40
        JSR DOCIO2
        JMP GET1
DOCIO
; THIS ROUTINE IS USED TO GET
; THE DISK DIR AND TO PRINT IT
; ENTER WITH X=DEVICE #
;           A=COMMAND
;           Y=BUFF LENGTH LSB
        STA ICCOM,X
        LDA # <BUFF
        STA ICBAL,X LSB OF ADDR
        LDA # >BUFF HI BYTE
        STA ICBAH,X
        TYA
DOCIO2
        STA ICBLL,X
        LDA #0
        STA ICBLH,X
        JSR CIOV
        RTS
;
;-----------------------------
; ROUTE THE OPTIONS
;-----------------------------
;
DOOPT
        LDA BUFF    GET 1ST CHAR
        STA OPT     SAVE OPTION
        CMP #'4     IS IT LOAD/SAVE?
        BCC ASKNAM  YES, IF <4
        BNE C1      ELSE 5 IF NOT 4
        JSR GETDIR  GET DIRECTORY
        JMP PRSKEY
C1
        RTS         RETURN W/OPT=5
ASKNAM
; RTS FM HERE RETURNS TO MAIN
; ROUTINE
;
        CMP #'2     SAVE OPTION ?
        BNE D1      NO
        LDA # <SFNAM ASK FOR NAME
        LDY # >SFNAM OF FILE TO SAVE
        JSR PRINT
        JMP GETNAM
D1
        LDA # <LFNAM
        LDY # >LFNAM
        JSR PRINT
GETNAM
; GET THE FILE NAME
        JSR PRTBLK
        LDA #9      POSITION CRSR
        STA COLCRS
        JSR CRSRON  TURN CRSR ON
        LDA #GETREC
        LDX #EDEV
        LDY #15     GET 15 CHAR ONLY
        JSR DOCIO
        JSR CRSROFF TURN CRSR OFF
        TYA
        BPL E1
        CPY #EOF    EOF ERROR ?
        BEQ E1      YEP, SO IT'S OK
        JMP ERROR   PRINT ERROR MSG
E1
        LDA BUFF
        CMP #'D     IS D 1/ST CHAR ?
        BEQ E2
        LDY #165    FILENAME ERROR
        JMP ERROR
E2
        LDA OPT     IS IT CHAR OR
        CMP #'3     SCRN FILE ?
        BNE SCRFIL  SCRN FILE IF <>3
;
;-----------------------------
; GET CHAR SET
;-----------------------------
;
CHRIO
        LDX #DDEV
        JSR DEVOFF
        LDY #READ
        JSR OPN
        TYA
        BPL GETSET
        JMP ERROR
GETSET
        LDA #GETCHR LOAD CHAR SET
        LDX #DDEV
        STA ICCOM,X
        LDA # <CHRSET
        STA ICBAL,X
        LDA # >CHRSET
        STA ICBAH,X
        LDA # <CHRSIZ
        STA ICBLL,X
        LDA # >CHRSIZ
        STA ICBLH,X
        JSR DOSCRN2
        LDY STAT
        JMP ERRTST  TEST FOR ERROR
;
;-----------------------------
; LOAD OR SAVE SCREEN FILE
;-----------------------------
;
SCRFIL
        LDX #DDEV   TURN OFF DISK
        JSR DEVOFF
        LDA OPT     GET OPTION
        CMP #'1     LOAD SCRN ?
        BNE F1      NO
        LDY #READ   YES
        BNE F2      ALWAYS
F1
        LDY #WRITE
F2
        JSR OPN     OPEN DISK DRIVE
        TYA
        BPL SCRIO
        JMP ERROR
SCRIO
        LDA OPT
        CMP #'1
        BNE B1
        LDA #GETCHR
        BNE B2      ALWAYS
B1
        LDA #PUTCHR
B2
        JSR DOSCRN
        STY STAT    SAVE ERR STATUS
;
;-----------------------------
; TEST FOR DISK I/O ERROR
; AND CONFIRM GOOD LOAD/SAVE
;-----------------------------
;
ERRTST
        TYA         TEST FOR ERROR
        BPL H2
        JMP ERROR
H2
        LDA OPT
        CMP #'2     SAVE OR LOAD ?
        BNE H3      LOAD IF #<> 2
        LDA # <SVOK VERIFY SAVE IS OK
        LDY # >SVOK
        JSR PRINT
        JMP PRSKEY
H3
        LDA # <LDOK
        LDY # >LDOK
        JSR PRINT   VERIFY LOAD IS OK
        JMP PRSKEY  GET NEXT OPTION
ERROR
; THIS ROUTINE PRINTS THE
; ERROR CODES ON SCREEN.
        LDX #2      CLR ERRR BUFFER
        LDA #SPACE  OF PREV ERRORS
I1
        STA ERRR+11,X
        DEX
        BPL I1
        JSR BIN2ASC CONVERT ERROR
        LDA #$9B
        LDY #12
        STA ERRR,Y  FORCE EOL
        JSR PRTBLK
        LDA # <ERRR
        LDY # >ERRR PRINT ERROR
        JSR PRINT
        JMP PRSKEY  GET NEXT OPTION
OPN
; THIS ROUTINE OPENS THE DISK
; DRIVE FOR LOAD/SAVE OF SCRNS
; OR CHAR SETS.
; ENTER WITH Y = AUX (R OR W)
        LDX #DDEV
        LDA #OPEN
        STA ICCOM,X
        LDA # <BUFF FILE NAME
        STA ICBAL,X
        LDA # >BUFF
        STA ICBAH,X
        TYA         Y = READ OR
        STA ICAX1,X WRITE AUX COMM.
        LDA #0
        STA ICAX2,X
        JSR CIOV
        STY STAT
        RTS
DOSCRN
; THIS ROUTINE LOADS OR SAVES
; THE SCRN.
; ENTER WITH ACC = PUTCHR OR
;            = GETCHR
;
        LDX #DDEV
        STA ICCOM,X
        LDA # <SCREEN BEG OF SCRN
        STA ICBAL,X
        LDA # >SCREEN
        STA ICBAH,X
        LDA # <SCRSIZ SIZE OF SCRN
        STA ICBLL,X
        LDA # >SCRSIZ
        STA ICBLH,X
DOSCRN2
        JSR CIOV
        STY STAT
        JSR DEVOFF
        RTS
BIN2ASC
; THIS ROUTINE CONVERTS THE Y
; REGISTER ERROR INTO DECIMAL
; ASCI AND PUTS IT IN THE
; I/O BUFFER.
; ENTER WITH Y=ERROR #
        STY STAT    SAVE ERROR
        TYA
        LDY #9      ERRR BUFF INDEX
        LDX #0
J1
        CMP #100    LESS THAN 100 ?
        BCC SAV100
        SBC #100    COUNT THE 100'S
        INX         COUNT # OF 100'S
        JMP J1
SAV100
        JSR SAVERR  CONVERT #
        LDX #0
K1
        CMP #10     COUNT THE 10'S
        BCC SAV10
        SBC #10     SUB TIL <10
        INX
        JMP K1
SAV10
        JSR SAVERR  CONVERT IT
        CLC
        ADC #$30    CONVERT 1'S
        STA ERRR,Y  PUT IN BUFFER
        INY
        RTS
SAVERR
        PHA
        TXA         GET # TO CONVERT
        ADC #$30    CONVERT IT
        STA ERRR,Y  PUT IN BUFFER
        INY
        PLA         RESTORE ACC
        RTS         DO TIL DONE
DEVOFF
; TURN OFF DEVICE
; ENTER WITH X=DEVICE #
        LDA #CLOSE  CLOSE COMMAND
        STA ICCOM,X
        JSR CIOV
        RTS
POS2412
        LDA #24     POS 24,12
        STA COLCRS
        LDA #0
        STA COLCRS+1
        LDA #12
        STA ROWCRS
        RTS
```

```
PRSKEY
        JSR PRTBLK
        LDA # <PRES TELL TO PRESS
        LDY # >PRES START KEY
        JSR PRINT
        LDA #6
L1
        CMP CONSOL  RETURN WHEN IT'S
        BNE L1      PRESSED
        RTS
PRTBLK
        LDA # <BLNK PRINT BLANK LINE
        LDY # >BLNK
        JSR PRINT
        RTS
CRSROFF
        LDX #1      TURN CRSR OFF
        STX CRSINH
        RTS
CRSRON
        LDX #0      TURN CRSR ON
        STX CRSINH
        RTS
CLSALL
; CLOSE ALL DEVICES EXC 0
        LDX #EDEV
        JSR DEVOFF
        LDX #SDEV
        JSR DEVOFF
        LDX #KDEV
        JSR DEVOFF
        LDX #DDEV
        JMP DEVOFF
DOS
        JMP ($0A)
        RTS
NODOS
; ANNOUNCE NO DOS ON DISK
        LDA # <NONE
        LDY # >NONE
        JSR PRINT
        JMP PRSKEY
;
;------------------------------
;DEVICE NAMES
;------------------------------
ECOLON
        .BYTE "E:",EOL
SCOLON
        .BYTE "S:",EOL
KCOLON
        .BYTE "K:",EOL
DIRNAM
        .BYTE "D:*.*",0
;
;------------------------------
; MESSAGES
;------------------------------
;
CRIT
        .BYTE 125,29,29,127,30,30,30,30
        .BYTE "SCREDIT (C) 1984 RON HODGE",EOL
OPT1
        .BYTE 29,29,127
        .BYTE "1- LOAD SCREEN",EOL
OPT2
        .BYTE 127,"2- SAVE SCREEN",EOL
OPT3
        .BYTE 127,"3- LOAD CHAR SET",EOL
OPT4
        .BYTE 127,"4- D1: DIRECTORY",EOL
OPT5
        .BYTE 127,"5- EXIT TO EDITOR",EOL
OPT6
        .BYTE 127,"6- EXIT TO DOS",EOL
QUES
        .BYTE 29,127
        .BYTE "WHICH OPTION ?        ",EOL
SFNAM
        .BYTE 29,127
        .BYTE "NAME OF FILE TO SAVE ?",EOL
LFNAM
        .BYTE 29,127
        .BYTE "NAME OF FILE TO LOAD ?",EOL
LDOK
        .BYTE 29,127
        .BYTE "LOAD COMPLETED",EOL
SVOK
        .BYTE 29,127
        .BYTE "SAVE COMPLETED",EOL
PRES
        .BYTE 127
        .BYTE "PRESS START TO CONTINUE",EOL
ERRR
        .BYTE 127
        .BYTE "ERROR:        ",EOL
NONE
        .BYTE 127
        .BYTE "ERROR: NO DOS ON DISK",EOL
BLNK
        .BYTE "                                        ",EOL
MSG
; ADDRESSES OF MESSAGES FOR
; PRINTING THE MENU
        .WORD QUES,OPT6,OPT5,OPT4,OPT3,OPT2,OPT1,CRIT
SPACES
        .BYTE 32,32,32,32,32
BUFF
        .BYTE 0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0
;
;------------------------------
; START OF SCREEN EDITOR
;------------------------------
;
INIT
        LDY #15     INIT ZERO PAGE
AA1
        LDA ZPAG,Y
        STA $F0,Y
        DEY
        BPL AA1
COLOR
        LDY #4
BB1
        LDA HUES,Y
        STA COLOR0,Y
        DEY
        BPL BB1
        LDA # >CHRSET
        STA CHBAS
CLRPM
        LDA #0
        LDY #$FF
CC1
        STA PB0,Y
        DEY
        BNE CC1
        STA SCBIT   SET HSCROL REG
        STA HSCROL
        STA MODFLG  0 = IN EDIT MODE
COLPL
        LDA #$5A
        STA PCOLR0  PINK CURSOR
PMPOS
        LDA #LFTPM
        STA HPOSP0
        STA HORP0
        LDA #0
        STA HPOSP0+1
        STA HPOSP0+2
        STA HPOSP0+3
SETPRI
        LDA #1      ALL PLAYRS,THEN
        STA GPRIOR  THEN ALL PF'S
        LDA #0
        STA SIZEP0  SET P0 DBL WIDTH
        LDA #WIDTH
        JSR DRWPM   DRAW CURSOR
SETPMB
        LDA # >PLBAS
        STA PMBASE
ENBL
        LDA #3
        STA GRACTL  ENABLE PM'S
CLRWND
        LDY #161    CLR TEXT WINDOW
        LDA #0
DD1
        STA $73FF,Y
        DEY
        BNE DD1
SETDISP
        SEI         TURN OFF INT
        LDA #0      TURN OFF SCRN
        STA SDMCTL
        LDY #47
EE1
        LDA SDLIST,Y
        STA DLIST,Y
        DEY
        BPL EE1
        LDA # <DLIST
        STA SDLSTL  CHNG DISP/SCRN
        LDA # >DLIST
        STA SDLSTH
        LDA #62     SINGLE LINE DISP
        STA SDMCTL
        LDA # <DLI  POINT TO OUR
        STA VDSLST  DLIST INTERRUPT
        LDA # >DLI
        STA VDSLST+1
        CLI
VBLSET
        LDY # <VBLRTN
        LDX # >VBLRTN
        LDA #7      DEFERRED VBL
        JSR $E45C
        LDA #$C0    ENABLE DLI'S
        STA NMIEN   AND VERT BLANKS
;
        JSR OPNKBD  OPEN KEYBOARD
        JMP MAIN
;
;------------------------------
; VERTICAL BLANK ROUTINE
;------------------------------
;
; VBL ROUTINE READS THE J/S
; AND SCROLLS THE SCREEN IF J/S
; IS MOVED.  THE SCROLL FLAG
; (SCFLG) IS SET IF SCROLLING
; IS UNDERWAY, SO THE MAIN
; ROUTINE SIMPLY LOOPS WHILE
; SCREEN IS SCROLLING.
;
VBLRTN
        LDA MODFLG  EXIT IF IN
        BNE FF2     MENU MODE
        LDX SCFLG   SCROLLING ?
        BEQ FF1     NO
        LDA #14     SLOW DOWN SCROL
        AND #1      TO EVERY OTHER
        BNE FF2     FRAME
        JSR SCROL
        JMP XITVBV
FF1
        LDA TRIG0   IS TRIG PRESSED?
        BEQ FF2     YES
        JSR CHKJS   READ J/S,ADJ SCFLG
FF2
        JMP XITVBV  END OF VBL !
;
;------------------------------
; MAIN ROUTINE
;------------------------------
;
MAIN
        LDA #14     JIFFY ELAPSED ?
GG1
        CMP #14
        BEQ GG1     NO
        LDA SCFLG   SCROLLING ?
        BNE MAIN    YES;SO LOOP BACK
CHKTRG
        LDA TRIG0   TRIGGER PUSHED ?
        BNE CHEK    NO, CHEK CONSOL
        LDA STICK0
        CMP #7      PUSHED RITE?
        BNE HH1
        JSR CRSRIT
        JMP WAIT
HH1
        CMP #11     LEFT ?
        BNE HH2
        JSR CRSLFT
        JMP WAIT
HH2
        CMP #14     UP ?
        BNE HH3
        JSR CRSUP
        JMP WAIT
HH3
        CMP #13     DOWN ?
        BNE CHEK
        JSR CRSDWN
        JMP WAIT
CHEK
        LDA CONSOL
        CMP #OPTION OPTION PRESSED ?
        BNE JJ1
        JSR CHGPRI  CHANGE PRIORITIES
        JMP CLIK
JJ1
        CMP #SELECT SELECT PRESSED ?
        BNE JJ2
        JSR SHWCHR  SHOW CHAR
        JSR SHWHEX  SHOW HEX VALUE
        JMP CLIK
JJ2
        CMP #START  START PRESSED ?
        BNE CHEKEY
        JSR FLIPCH  FLIP CHAR SETS
CLIK
        LDA #4
        STA CONSOL  CLIK
        JMP WAIT    WAIT1
CHEKEY
        LDA CH      GET KEY
        CMP #$FF    ANY PRESSED ?
        BEQ KK4     NO
KK1
        CMP #ESC    ESCAPE KEY ?
        BNE KK2     NO
        JMP CHGMOD  YEP,GOTO MENU
KK2
        JSR CMPCHR  PRINTABLE CHAR ?
        BCS KK4     NOT IF CARRY SET
        JSR TSTARW  CRSR CHAR ?
        BCS KK5     YES,IF CARRY SET
        JSR GETCH   ELSE GET CHAR
        JSR CONVERT CHG ASCI TO DISP
        JSR PUTCHAR AND PUT ON SCRN
        JMP WAIT
KK4
        JMP MAIN
KK5
```

```
        LDA #$FF
        STA CH
        JMP CLIK
WAIT
        LDA #$FF        CLR THE KEYBOARD
        STA CH
        LDX #4          # OF JIFFIES
LL1
        LDA $14         GET FRAME COUNT
LL2
        CMP $14
        BEQ LL2
        DEX             WAIT TIL 4 FRAMES
        BNE LL1         HAVE ELAPSED
        JMP MAIN        THEN LOOP TO MAIN
;
;-------------------------------
; PROCESS KEYBOARD
;-------------------------------
;
GETCH
; THIS ROUTINE ACCEPTS INPUT
; FROM THE KEYBOARD
        LDX #KDEV
        LDA #0          BUFFER LENGTH=0
        STA ICBLL,X     SO CHAR RETURNS
        STA ICBLH,X     IN ACC
        LDA #GETCHR
        STA ICCOM,X
        JSR CIOV
        RTS
CONVERT
; CONVERT ATASCI TO INTERNAL
; CODE. ENTER W/ATASCI IN ACC
        TAY
        ROL A
        ROL A
        ROL A
        ROL A
        AND #3
        TAX
        TYA
        AND #$9F
        ORA AT2INT,X
        STA CHR         SAVE INT CODE
        RTS
PUTCHAR
        JSR SETMP
        LDY #0
        LDA CHR
        STA (TEMP0),Y
;
;-------------------------------
; MOVE CURSOR WITH/WRAP AROUND
;-------------------------------
;
CRSRIT
        LDY COL
        CPY #COLM       ON RITE COLUMN ?
        BEQ MM1         IS IT = OR > ?
        BCS MM1         YEP
        INC COL         NO, UPDATE COL
        INC SCRN        AND ADDR UNDER
        BNE MM3         CURSOR
        INC SCRN+1
MM3
        LDA HORP0       MOVE CURSOR
        CLC
        ADC #HOR
        STA HORP0
        STA HPOSP0
        RTS
MM1
        LDA #0
        STA COL         WRAP CURSOR
        LDA #LFTPM      PUT IN LEFTMOST
        STA HORP0       POSITION
        STA HPOSP0
        LDA SCRN        CHANGE SCRN PTR
        SEC
        SBC #19
        STA SCRN
        BCS MM2
        LDA SCRN+1
        SBC #0
        STA SCRN+1
MM2
        RTS
CRSLFT
        LDY COL         ON LEFT COLUMN ?
        BEQ NN1         YES
        DEC COL         NO, JUST DEC IT
        LDA SCRN        AND SCRN POINTER
        BNE NN2
        DEC SCRN+1
NN2
        DEC SCRN
        LDA HORP0       MOVE CURSOR
        SEC
        SBC #HOR
        STA HORP0
        STA HPOSP0
        RTS
NN1
        LDA #19
        STA COL         WRAP AROUND CURSOR
        CLC
        ADC SCRN
        STA SCRN
        LDA SCRN+1
        ADC #0
        STA SCRN+1
        LDA #RITPM
        STA HORP0
        STA HPOSP0
        RTS
CRSUP
        LDY ROW         ON TOP ROW ?
        BEQ PP1         YES
        DEC ROW         NO
        DEC SCRN+1      MOV UP 2 PAGES
        DEC SCRN+1
        LDA #0          ERASE CRSR
        JSR ERPM
        STY VOFF
        LDA #WIDTH
        JSR DRWPM
        RTS
PP1
        LDA #ROWM
        STA ROW         WRAP AROUND CURSOR
        LDA SCRN+1      CHANGE SCRN PTR
        CLC
        ADC #18
        STA SCRN+1
        LDA #0
        JSR ERPM        ERASE CRSR
        LDY #BOTPM
        STY VOFF
        LDA #WIDTH
        JSR DRWPM
        RTS
CRSDWN
        LDY ROW         ON BOTTOM ROW ?
        CPY #ROWM
        BEQ QQ1
        BCS QQ1         YES
        INC ROW         NO
        INC SCRN+1      MOV DWN 2 PAGES
        INC SCRN+1
        LDA #0
        JSR ERPM
        LDA VOFF
        CLC
        ADC #HITE
        STA VOFF
        LDA #WIDTH
        JSR DRWPM
        RTS
QQ1
        LDA #0
        STA ROW         WRAP AROUND CURSOR
        LDA SCRN+1      CHANGE SCRN PTR
        SEC
        SBC #18         ROWS*2
        STA SCRN+1
        LDA #0
        JSR ERPM        ERASE CRSR
        LDY #TOPPM
        STY VOFF
        LDA #WIDTH
        JSR DRWPM
        RTS
;-------------------------------
; READ JOYSTICK FOR SCROLLING
;-------------------------------
CHKJS
        LDA STICK0      GET J/S
        CMP #7          PUSHED RITE ?
        BNE RR1
        JSR CHKLF       CAN WE SCROL LFT ?
        BCS RR2         NO
        LDA #8
        STA SCBIT
        LDA #$80
        STA SCFL0
        RTS
RR1
        CMP #11         PUSHED LEFT ?
        BNE RR2
        JSR CHKRT       CAN WE SCROL RIT ?
        BCS RR2
        LDA #0
        STA SCBIT
        LDA #20
        STA SCFL0
RR2
        RTS
ERPM
DRWPM
; ENTER WITH ACC = 0 TO ERASE
; OR ACC = WIDTH TO DRAW CRSR
        LDY VOFF
        LDX #HITE
SS1
        STA (PMLO),Y
        DEY
        DEX
        BNE SS1
        RTS
;
;-------------------------------
; FINE SCROLL A BYTE AT A TIME
;-------------------------------
;
CHKLF
        LDY SCRLO       TEST FOR EAST MAX
        CPY #225        IS LOBYTE >225 ?
        BCC TT1         NO
        LDA SCRHI       YES;IS HYBYTE=0?
        BNE TT2         NO ; DON'T SCROL
TT1
        INC SCRLO
        BNE TT3
        INC SCRHI
TT3
        CLC
        RTS
TT2
        SEC
        RTS
CHKRT
        LDY SCRLO
        CPY #6
        BCC UU1         CHEK HI BYTE IF <6
        DEC SCRLO
        CLC
        RTS
UU1
        LDX SCRHI
        BEQ UU2         DONT'T SCROL IF 0
        DEC SCRLO       ELSE DEC LO CTR
        BPL UU3
        DEC SCRHI
UU3
        CLC
        RTS
UU2
        SEC
        RTS
SCROL
        TXA
        BPL RITSC       RITE SCROL
LFTSC
        DEC SCBIT       DEC SCROL BIT
        BEQ LFTLP
        LDA SCBIT
        STA HSCROL
        RTS
LFTLP
        LDX #28         PT TO HIEST LO LMS
V1
        INC DLIST+3,X
        BNE V2
        INC DLIST+4,X ELSE INC HIBYT
V2
        DEX             DLI INS
        DEX             LMS HI
        DEX             LMS LO
        BPL V1
        LDA #0
        STA SCFL8
        INC SCRN        UPDATE SCREEN
        BNE V3
        INC SCRN+1
V3
        LDA #8
        STA SCBIT
        STA HSCROL
        RTS
RITSC
        INC SCBIT
        LDY SCBIT
        CPY #8
        BCS RITLP
        LDA SCBIT
        STA HSCROL
        RTS
RITLP
        LDX #28
WW1
        LDA DLIST+3,X RESET LO BY
        BNE WW2
        DEC DLIST+4,X DEC HIBYTE
WW2
        DEC DLIST+3,X
        DEX             DLI
        DEX             LMS HI
        DEX             LMS LO
        BPL WW1
        LDA #0
        STA SCFL8
        LDA SCRN
        BNE WW3
        DEC SCRN+1
WW3
        DEC SCRN
        LDA #0
        STA SCBIT
```

```
        STA HSCROL
        RTS
;
;------------------------------
; MISC ROUTINES
;------------------------------
;
CHGPRI
        LDA #1
        CMP GPRIOR
        BNE XX1
        LDA #4
        STA GPRIOR
        RTS
XX1
        STA GPRIOR
        RTS
SHWHEX
; DISP CHAR UNDER CRSR IN HEX
        LDY #0
        LDA (TEMP0),Y
        PHA
        LSR A         MOV LEFT NIB TO RITE
        LSR A
        LSR A
        LSR A
        JSR HEX
        PLA
        AND #$0F      CONVERT LEFT NIBBLE
HEX
        CLC
        ADC #16
        CMP #26
        BCC YY1
        CLC
        ADC #7
YY1
        STA HXTXT,Y
        INY
        RTS
DLI
; THIS DISPLAY LIST INTERRUPT
; OCCURS JUST BEFORE THE FIRST
; TEXT WINDOW MODE LINE.
        PHA
        LDA #$E0      RESTORE CHAR SET
        STA WSYNC
        STA CHBASE
        LDA TXTCLR    TXT WINDOW COLOR
        STA COLPF2    PUT IN HDWR REG.
        PLA
        RTI
;
SETMP
; THIS COMPENSATES FOR SHIFT
; OF CHAR BY SCROL REG
        LDA SCRN      COPY PTR
        STA TEMP0
        LDA SCRN+1
        STA TEMP1
        LDA SCBIT
        BEQ ZZ3       XIT IF NO ADJ NEC
        LDA TEMP0     8, SO INC PTR
        BNE ZZ2
        DEC TEMP1
ZZ2
        DEC TEMP0
ZZ3
        RTS
SHWCHR
        JSR SETMP
        LDY #0
        LDA (TEMP0),Y
        AND #63
        STA CHDIS
        RTS
SDLIST
        .BYTE $70,$70,$70 ; 3*8 BLNK LINES
        .BYTE $57,$00,$60 ; 1 LINE GR.2
        .BYTE $57,$00,$62 ; 2
        .BYTE $57,$00,$64 ; 3
        .BYTE $57,$00,$66 ; 4
        .BYTE $57,$00,$68 ; 5
        .BYTE $57,$00,$6A ; 6
        .BYTE $57,$00,$6C ; 7
        .BYTE $57,$00,$6E ; 8
        .BYTE $57,$00,$70 ; 9
        .BYTE $D7,$00,$72 ; 10 W/DLI
        .BYTE $42,$00,$74 ; TEXT WINDOW
        .BYTE $42,$28,$74
        .BYTE $42,$50,$74
        .BYTE $42,$78,$74
        .BYTE $41   ; JVB
        .WORD DLIST
HUES
        .BYTE 40,202,148,70,0
ZPAG
; THIS TABLE LOADS ZERO PAGE
; VARIABLES
        .BYTE 8       SCBIT SCROL BIT CTR
        .BYTE $FF     CHR   DISP CHAR
        .BYTE 2       SCRN  PTR TO SCRN
        .BYTE $60     SCRN+1
        .BYTE 5       SCRLO  SCRL BYTE CTR
        .BYTE 0       SCRHI   SCRL HI
        .BYTE  <PB0   PMLO PTR TO PM0
        .BYTE  >PB0   PMHI    "
        .BYTE 0       COL     CURSOR COLUMN
        .BYTE 0       ROW       "     ROW
        .BYTE 0       TEMP0
        .BYTE 0       TEMP1
        .BYTE TOPPM   VOFF VRT OFFSET P0
        .BYTE LFTPM   HPOSP0 SHADOW
        .BYTE 0       SCFLG <>=SCROLLING
        .BYTE 148     TXTCLR TXT WIN COLR
FLIPCH
; FLIP CHAR SETS
        LDA # >CHRSET
        CMP CHBAS
        BEQ A1A
        STA CHBAS
        RTS
A1A
        LDA #$E0
        STA CHBAS
        RTS
CMPCHR
        PHA
        AND #$3F      STRIP HI NIBBLE
        LDY #5
BA1
        CMP BADCH,Y
        BEQ BA2
        DEY
        BPL BA1
        PLA
        CLC
        RTS
BA2
        PLA
        SEC           PRINTABLE CHAR
        RTS
BADCH
; KEYS TO WEED OUT AND IGNORE
        .BYTE RETURN,ESC,TAB,DEL,INSERT,CLR
TSTARW
; TEST FOR CONTOL ARROW CHAR
        CMP #$8F      RETURN W/CARRY
        BNE TS1       SET IF CRSR MOVED
        JSR CRSDWN
        SEC
        RTS
TS1
        CMP #$8E
        BNE TS2
        JSR CRSUP
        SEC
        RTS
TS2
        CMP #$86
        BNE TS3
        JSR CRSLFT
        SEC
        RTS
TS3
        CMP #$87
        BNE TS4
        JSR CRSRIT
        SEC
TS4
        RTS
CHGMOD
; THIS ROUTINE EXITS TO THE
; MENU.
        LDA #$FF      TURN OFF VBL
        STA MODFLG
        LDX #KDEV     TURN OFF KBOARD
        JSR DEVOFF
        LDA #0
        STA GRACTL    TURN OFF PM'S
        STA HPOSP0
        LDA #34
        STA SDMCTL
        STA DMACTL
        JMP MENU
AT2INT
; ATASCI TO INTERNAL CONVERSION
        .BYTE $40,$00,$20,$60
        .END
```

SpartaDOS and ARCHIVER II™



From the authors of the *CHIP!*

SpartaDOS is our all new command driven DOS fully compatible with all disk drives made for the Atari Home Computers.™ Supports all Atari compatible densities - both single and double sided, 5¼ and 8 inch.

UltraSpeed™ I/O allows read/write 2 to 4 times faster than standard!* Comprehensive utilities include: a sector copier, RS-232 handler, and DOS file translators.

ARCHIVER II allows UltraSpeed backup of all diskettes and compacting of multiple programs on each disk. Includes an exceptional editor, a speed check, and is fully automatic.

Available now for Atari 810 and 1050 drives! Coming soon for most other brands.

Suggested Retail Price $129.95 (plus shipping and installation)

See SpartaDOS and ARCHIVER II now at leading computer stores or order direct from:

Spartan Software
Div. of ICD, Inc.
828 Green Meadow Ave.
Rockford, IL 61107
815-229-2999 (1:00-9:00 p.m. CST)

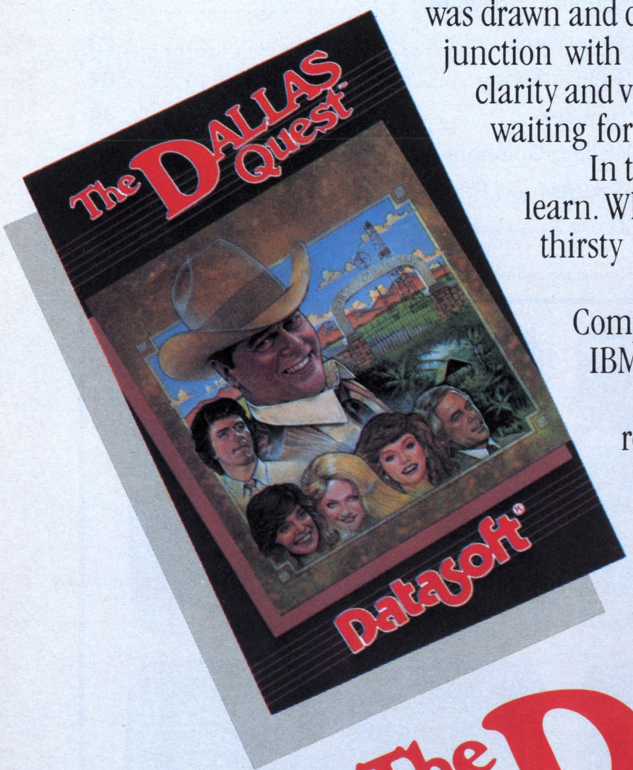
*Speed depends on drive hardware. A chip replacement is required for most drives.

HOW FAR WOULD YOU GO TO BEAT J.R.™ AT HIS OWN GAME?

This year's hottest graphic adventure game puts you in the hot seat. If you're like most of us, you've probably sat in front of a television and cooled your heels watching J.R.™ walk all over family, friends, anyone who gets in his way.

Now it's your turn to even the score. The Dallas Quest™ lets you write yourself into the script. And out of the country. The adventure takes you to hidden jungles deep in South America where primitive gods rule the land. Then back to Southfork™ where money reigns supreme. As many as 40 scene changes over 2 continents test your logic, determination, grit and eventually greed. If you succeed in outwitting J.R.™ by securing a secret oil field for Sue Ellen, there's $2,000,000 and her personal congratulations waiting for you.

From the opening to the closing scene you'll be captivated by the graphic realism. Each one was drawn and detailed by professional artists and developed in conjunction with the producers of "Dallas."™ The hi-resolution clarity and visual panning motions are only a few of the surprises waiting for you.



In the Dallas Quest,™ there's one thing you're certain to learn. Whether pursued by the power hungry J.R.,™ or blood thirsty natives, it really is a jungle out there.

Available now for Atari and Commodore 64 Computers and coming soon for the Apple II Series and IBM PC and PC/JR. Suggested retail $34.95.

Check with your local home computer software retailer for The Dallas Quest,™ and to learn of other great programs from Datasoft® send for a free consumer catalog.

The DALLAS Quest™

By Datasoft®

19808 Nordhoff Place, Chatsworth, CA 91311 Phone (818) 701-5161

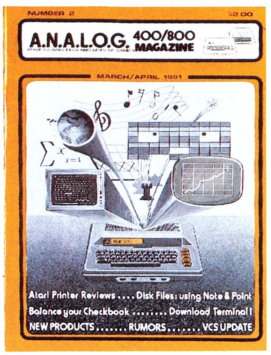
CIRCLE #153 ON READER SERVICE CARD.

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

BACK ISSUES

Catch up on what you've missed!



ISSUE 2
Wasting Arrays
Atari's CPU
Download Terminal
Converting BASIC
Disk Files

ISSUE 8
GTIA Graphics
Audio in Your Programs
NOREM
Graphic Violence
Color Slot Machine

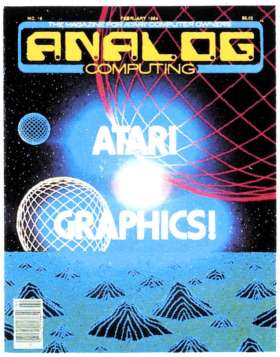
ISSUE 9
Build Your Own 400 Keyboard
Harvey Wallbanger
Forth-Dos
Letter Writer

ISSUE 11
Strings in BASIC
C:CHECK
Disk Cataloging Utility
Adventure in the Fifth Dimension
Moving Missiles in BASIC

ISSUE 13
Fine Scrolling Part 1
Roundup
Space Assault
Observational Astronomy
CIO Routines

ISSUE 14
Fine Scrolling Part 2
Disassembler in BASIC
Hexpad
Lumberjack
Retrofire!

ISSUE 15
Fine Scrolling Part 3
Knights and Chalices
Music Synthesizer
Bricklayer's Nightmare
Alternative Keyboard Handler



ISSUE 16
Fine Scrolling Part 4
Create-A-Font
Bar Chart Subroutine
Shooting Stars
3-D Object Rotation

All back issues $4.00 each

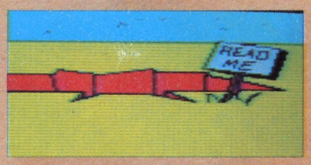
Send check or money order to:
ANALOG Back Issues
P.O. Box 615
Holmes, PA 19043

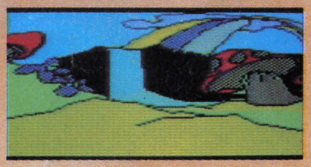
MasterCard and Visa orders call:
1-800-345-8112
in PA. 1-800-662-2444

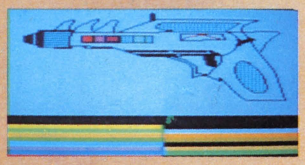
S.A.G.E.

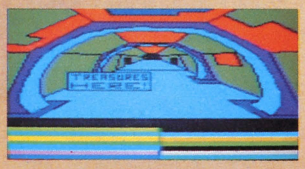
SCOTT ADAMS'

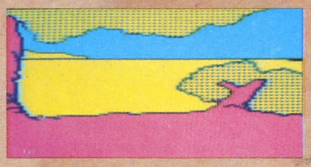
GRAPHICS EDITOR

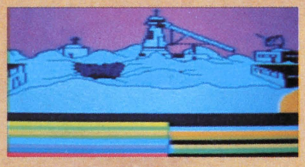












An Atari computer doubling as an art studio? You bet! Introducing S.A.G.E. — the Scott Adams Graphics Editor.

S.A.G.E. lets you draw and edit highly compressed pictures that use over 50 different colors and shades. And S.A.G.E. was designed with simplicity in mind — nearly every command can be executed with a single key press or a simple push/pull of the joystick.

Programming buffs will love the ease with which S.A.G.E. pictures can be drawn and then placed into their own Basic or machine-language programs. And the hi-res pictures you create will look the same whether your Atari has a GTIA chip or an older CTIA chip.

The S.A.G.E. package includes both the 48K tape and disk versions. You also receive a clear, comprehensive instruction manual that every "artist" in the family can easily use.

S.A.G.E. - only $49.95! Draw your own conclusion — see your dealer today or call our toll-free number to place your order. Don't forget to ask for our free catalog!

SAGE-Scott Adams' Graphics Editor

Visit your local dealer today, or call us for your copy!

800-327-7172

Requires an Atari computer with 48K. Both TAPE and DISK are in one package.

054-0166 . . . . . $49.95

Adventure® INTERNATIONAL

BOX 3435 • LONGWOOD, FL 32750

(305) 862-6917

CIRCLE #150 ON READER SERVICE CARD.